$2.00

kilobaud™

The Small Computer Magazine

ISSUE #4 April 1977

Articles

Interrupts Exposed . . . *using microprocessor interrupt capability effectively* . . . *Dan La Dage* 18
Clocked Logic . . . *Part 2: Some Basic Applications* . . . *Don Lancaster* 22
Build an Eight Channel Multiplexer for Your Scope . . . *W. J. Prudhomme* 28
Sorting Routines . . . *explanation of common sorting techniques* . . . *Andrew J. Rerko* 34
Number Rounding Program . . . *simplifying the decimals* . . . *Jack A. Inman* 40
Meet the Tarbell/KC Interface . . . *Don Tarbell* 44
Super-Tester . . . *a digital design aid* . . . *Morris Krieger* 50
The Hobbyist's Operating System . . . *Part 3: Command Language Processing* . . . *Dick Wilcox* 54
The Slow-Stepping Debugger . . . *Howard R. Bendrot* 60
BASIC — The Easy Way . . . *Gabriel F. Gargiulo* 64
Now You Can Use Software Timing Loops . . . *Tim Barry* 68
KIM-1 Memory Expansion . . . *adding memory to this popular system is a snap* . . . *Bob Haas* 74
Heavy Duty Power Supply . . . *juice for an entire system* . . . *William Cattey* 78
Digital Audio . . . *the next revolution in sound?* . . . *Tom Scott* 82
HI-LO . . . *impress your friends when they visit* . . . *Jim Huffman* 88
Interfacing the Analog World . . . *Dr. Douglas Hogg* 90
Everything about Semiconductor Memory . . . *at least 4K is needed for BASIC* . . . *Peter A. Stark* 96
Three-State Logic . . . *explanation of a key microprocessor element* . . . *John Molnar* 106
Automatic Memory Dumper . . . *utility dump program for 6800 users* . . . *Jim Huffman* 110
Hangmath! . . . *a new puzzle/game* . . . *Phil Feldman, Tom Rugg* 112
Now — BASIC for the 8008 — Even! . . . *Grant Runyan* 116
Microprogramming . . . *an insight into microprocessor design* . . . *Dr. Lance A. Leventhal* 120
Computerized Babysitter . . . *graphic display for kids (and parents)* . . . *Alfred S. Baker* 130

Features

Publisher's Remarks . . . 2
Editor's Remarks . . . 4
Books . . . 6
Lookahead . . . 8
Around the Industry . . . 10
The BASIC Forum . . . 12
News of the Industry . . . 14
Letters . . . 16
Corrections . . . 46
Glossary . . . 124

Old Fashion VALUE in the U.S.A. Tradition

Our 6800 computer system represents the best value available today, with no sacrifice in performance.

I would like to explain why this is true. The most basic reason is that the 6800 is a simpler, more elegant machine. The 6800 architecture is memory oriented rather than bus oriented as are the older 8008, 8080 and Z-80 type processors. This is an important difference. It results in a computer that is far easier to program on the more basic machine language and assembly language levels. It also results in a far simpler bus structure. The 6800 uses the SS-50 bus which has only half the connections needed in the old S-100 (IMSAI/MITS) bus system. If you don't think this makes a difference, take a look at the mother boards used in both systems—compare them. The SS-50 system has wide, low impedance 0.1 lines with good heavy, easily replaced Molex connectors. The S-100 bus, on the other hand, has a very fine hair-like lines that must be small enough to pass between pins on a 100 contact edge connector. I'll give you one guess which is the most reliable and noise free. As for cost—well any of you who have purchased extra connectors for your S-100 machines know what kind of money this can run into. The 6800 is supplied with all mother board connectors. No extras, or options like memory, or connectors for the mother board are needed in our 6800 system.

The 6800 is not beautiful, but "Oh Boy" is it functional. That plain black box is strong and it has an annodized finish. This is the hardest, toughest finish you can put on aluminum. Most others use paint, or other less expensive finishes. The 6800 does not have a pretty front panel with lights and multicolor switches. This is because the lights and switches are not only expensive, and unnecessary, but also a great big pain to use. We don't crank up the 6800; we use an electric starter—a monitor ROM called Mikbug. He automatically does all the loading for you without any time wasting switch flopping. So in the 6800 system you don't buy something expensive (the console) that you will probably want to stop using as soon as you can get your hands on a PROM board and a good monitor.

That's another thing. Mikbug® is a standard Motorola part. It is used in many systems and supported by the Motorola software library in addition to our own extensive collection of programs. It is not an orphan like many monitor systems that are unique to the manufacturer using them and which can only run software provided by that manufacturer. Check the program articles in Byte, Interface and Kilobaud. You will find that almost all 6800 programs are written for systems using a Mikbug® monitor. Guess how useful these are if you have some off-brand monitor in your computer.

The 6800 will never win any beauty prizes. It is like the Model "T" and the DC-3 not pretty, but beautiful in function. It is simple, easy to use and maintain and does its job in the most reliable and economical way possible. What more could you want?

Mikbug ® is a registered trademark of Motorola Inc.

SWTPC 6800
Computer System
with serial interface and 4,096 words of memory $395.00

☐ Enclosed is $395 for my SwTPC Computer Kit ☐ Send Data

☐ or BAC_____ #_____

☐ or MC_____ Ex Date_____

NAME

ADDRESS

CITY STATE ZIP

Southwest Technical Products Corp., Box 32040, San Antonio, Texas 78284

THE LOGICAL CHOICE—First in a series

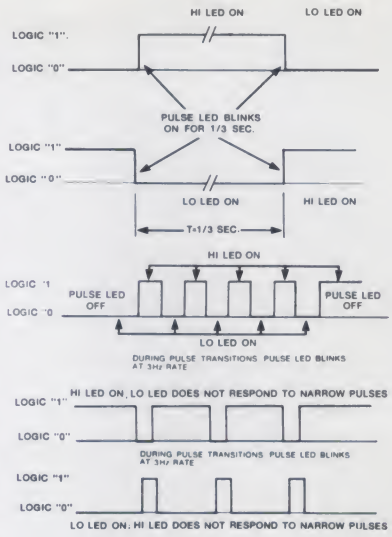
Logic Probe 1 is a compact, enormously versatile design, test and troubleshooting tool for all types of digital applications. By simply connecting the clip leads to the circuit's power supply, setting a switch to the proper logic family and touching the probe tip to the node under test, you get an instant picture of circuit conditions.

LP-1's unique circuitry—which combines the functions of level detector, pulse detector, pulse stretcher and memory—makes one-shot, low-rep-rate, narrow pulses—nearly impossible to see, even with a fast scope—easily detectable and visible. HI LED indicates logic "1", LO LED, logic "0", and all pulse transitions—positive *and* negative as narrow as 50 nanoseconds—are stretched to ⅓ second and displayed on the PULSE LED.

By setting the PULSE/MEMORY switch to MEMORY, single-shot events as well as low- rep-rate events can be stored indefinitely.

While high-frequency (5-10MHz) signals cause the "pulse" LED to blink at a 3Hz rate, there is an additional indication with unsymmetrical pulses: with duty cycles of less than 30%, the LO LED will light, while duty cycles over 70% will light the HI LED.

In all modes, high input impedance (100K) virtually eliminates loading problems, and impedance is constant for all states. LP-1 also features over-voltage and reverse-polarity protection. Housed in a rugged, high-impact plastic case with strain-relieved power cables, it's built to provide reliable day-in, day-out service for years to come.



CSC'S MULTI-FAMILY LOGIC PROBE 1. AT $44.95, IT DIGS UP A LOT OF INFORMATION WITHOUT BURYING YOUR BUDGET.

HI/LO LED's—Display level (HI-logic "1", LO-logic "0") of signal activity at node under test

PULSE LED—Lets you know what's going on—and off. Indicates positive *and* negative pulse and level transitions. LP-1 stretches pulses as narrow as 50 nanoseconds to full ⅓ sec. (3Hz pulse rate)

PULSE/MEMORY Switch— PULSE position detects and stretches pulses as narrow as 50 nanoseconds to ⅓ sec. Switch to MEMORY and it stores single shot and low-rep-rate events indefinitely; HI/LO LED's remain active

Logic Family Switch— TTL/DTL or CMOS matches Logic "1" and "0" levels, for greater versatility. High Input Impedance—100K virtually eliminates circuit loading problems and is constant in both "0" and "1" states. CMOS position also compatible with HTL, HiNIL and MOS logic

Non-corrosive nickel-plated probe tip and clip leads—For reliable contacts and maximum life

Rugged high impact plastic case—Built to take it . . . in the lab or in the field

Protected—Features built-in reverse polarity and over-voltage protection; strain-relieved power cable

$44.95 Price tag—Costs so little it can be *your* personal property

CONTINENTAL SPECIALTIES CORPORATION

EASY DOES IT

44 Kendall St., Box 1942 New Haven, CT 06509
TWX: 710-465-1227
West Coast office: Box 7809, San Francisco, CA 94119
TWX: 910-372-7992

See your CSC dealer or call 203-624-3103 (East Coast) or 415-421-8872 (West Coast) 9 AM to 5 PM local time. Major credit cards accepted. Add $2.50 for shipping and handling in the U.S. and Canada on direct orders of $50.00 or less; $3.00 for orders over $50.00. On all foreign orders add 15% to cover shipping and handling.

I Print "Publisher's Remarks" 10 No End Run

Wayne Green

THOSE COMPUTERFESTS

It all started with a microcomputerfest at Trenton College last summer. The exhibitors were strung out along a narrow and dark hallway – plus a couple of lab rooms – with maybe a thousand hobbyists trying to move along the hall, to see the exhibits, to hear talks. It was chaos . . . and it was fun.

Next came Atlantic City in August with about 80 exhibitors and perhaps 3000 hobbyists . . . no one knows for sure how many attended and the show organizers have clammed up about it. The hotel was an epic in awful and the exhibit area was a steaming sauna during much of the show . . . tacky was the motif. Few of the exhibitors ever want to see Atlantic City again.

Hmmm . . . let's see . . . 3000 times $7.50 plus 80 times $400 . . . quite a few people did that arithmetic and came up with the answer: Let's put on computer hobby shows and get rich! A profusion of hobby shows were announced, all with one thing in common: They would be run by professional trade show firms since the entrepreneurs didn't know how to do it themselves. Professionals, alas, are expensive and little exhibitor enthusiasm developed.

For some reason there is a difference between a club run show and a professionally run event. The professionally run shows I've seen have been bummers, with all the fun of an undertaker's convention. You've probably been to some of these sterile trade shows.

There's another difference. While most club run events charge from $50 to maybe $200 for a booth, the professional shows can go over $1000 for a booth! The computer hobby industry is made up almost entirely of small, struggling firms and they don't need $1000 booth fees . . . or even $500 fees.

What about the computer hobbyist? What will be the best for him? The ideal computerfest would have every firm in the industry exhibiting so hobbyists could see and try out everything there is. Hobbyists tell us that they intend to spend some $2000 this year on their hobby and that means that just about every one of them is intensely interested in the hardware and software which is available. Thus the more firms kept away from a show, the less there is for the attendees.

Making a show too expensive for small firms to exhibit limits the value of the show to both the industry and to hobbyists. Having more than one show on a weekend does the same. Having too many shows also limits the possibilities for small firms to exhibit. Their resources are limited and they can't afford more than maybe four or five shows a year.

The biggest hamfest in the country is at Dayton. It is scheduled for April 29, 30, May 1, this year. Last year there were quite a few computer firms exhibiting and they sold equipment like crazy. There will be even more this year. After all, the *73 Magazine* readers have been reading computer articles for a solid year (some 300 pages of 'em) and this means there are tens of thousands of hams who are thinking about computers and who are interested in getting started with them. With about 15,000 hams expected at Dayton this year the computer exhibitors should do very well indeed.

When I found out that both the Trenton Club and *Personal Computer Magazine* had scheduled computer shows for the same weekend as Dayton, I got bent out of shape. I felt this would be a bad scene for both the exhibitors and the hobbyists since it would force firms to choose between the three shows. Everyone would lose . . . the exhibitors would lose the sales they might have made at the other shows and the hobbyists would get to see only a small part of the equipment they wanted to see.

Personal Computing agreed with this philosophy and, though it caused massive problems, moved their Philadelphia show date up a month to May 28-29th. Trenton wouldn't budge, so they will be running their show the same weekend as Dayton.

Looking on the bright side, the conflict seems likely to bring about a long range benefit to the industry in that there is now a move afoot to form a group which would "authorize" computerfests. This would have the effect of setting up a few big computerfests where hobbyists would be able to see the products of virtually all manufacturers. This would make it possible for small firms to show to the most advantage and at a lower cost than trying to display at a dozen shows all around the country.

SYSTEM DEMONSTRATIONS

One concept I've been trying to promote is that of setting up a schedule for manufacturers to demonstrate and answer questions about their equipment at computerfests. One or two minutes in a booth, looking at a computer, doesn't answer much for us . . . and most of us hate to make big dollar decisions on equipment on the basis of throwing a dart. We want to know everything we can find out about the hardware available . . . we'd like to see it working . . . ask questions . . . and find out how it shapes up compared to other systems.

Three computerfests have agreed to this concept so far and these should be of particular interest to hobbyists. These are the Atlanta show June 18-19th, the Seattle show July 30-31, and the Des Moines show Aug 21.

Technical talks where the people who know the equipment the best are able to demonstrate it and explain it would seem to be the most beneficial for hobbyists . . . and for the industry. You'll find me favoring this type of computerfest.

GROUND AXE

Though it is a difficult moral tussle, I try not to let the fact that a particular convention committee has shafted me for intruding too much with my considered advice on the value of shows. Dayton has, I understand, had a long term policy of putting me down . . . I still encourage readers and industry to go to the Dayton Hamvention. The chaps at PC76 in Atlantic City shafted me very thoroughly, but that has in no way influenced my negative views of their show. I have a file of over two dozen complaints from exhibitors to back up my beefs on that mess.

HELP!

Some pains are a pleasure to endure . . . like the growing pains of *Kilobaud*. Apparently we have our finger on the pulse of the hobbyist because things have been going great guns. Oh, we could use a few more good fundamental articles . . . more programs . . . more material on newer systems and products . . . more solutions to the myriad of problems which face the pioneer in this field.

The fact is that even though we've been expanding our staff, we still need more people to help us keep up with everything. When we get more articles we will need editors and production people to get them ready for publication . . . draftsmen to prepare the schematics . . . artists to paste up the type and artwork . . . and so forth.

We do have an immediate need for an advertising sales assistant. More articles will mean we can enlarge the magazine and this will in turn mean a need for more advertising to pay the printing bills for the larger magazine. A good background in microcomputers won't hurt if you're interested in this job.

We also want to expand our work in the *Kilobaud* lab where we are setting up and running most of the popular microcomputer systems . . . this means a technician who

knows what he's doing and can get various gadgets to talk to each other . . . coax recalcitrant equipment to perk . . . and help us test out programs on the various systems. A dedicated hobbyist could hardly find a more enjoyable way to "work."

Kilobaud is located in a small town (about 3500) in southern New Hampshire. The summers here are fantastic . . . the winters even better, with several ski areas nearby . . . lots of fishing, hunting, mountains to climb, and the beauty of a state whose main industry is vacations. The town is only an hour out of greater Boston . . . has the second largest A&P in New England . . . has the famed MacDowell Artists Colony . . . and probably is the most attractive small town in the state. Speaking of the state, there are no income taxes here, nor any sales tax . . . the taxes are about the lowest in the country and New Hampshire intends to keep 'em that way.

You'll be working with a staff of over 60 people who put *Kilobaud* and *73 Magazine* together . . . prepare the books . . . print 'em . . . and do all of the hundreds of things it takes to make a publishing house successful. A small business such as this is a superb place to learn about publishing . . . or to get in on the ground floor of the coming microcomputer explosion. You can bet that KB will have all sorts of irons in the fire as this field grows . . . and pieces of the action for those who rate it.

The plant is like nothing you've seen before, with everyone working in a large old (well over 200 years) house . . . over 40 rooms of beehive activity.

If this appeals to you . . . and you think you are undoubtedly the best possible person for the job . . . convince us.

THE *KILOBAUD* STAFF

What does it take to get a magazine published? The fact is that it takes quite a crew to do the job right.

Just to follow an article through the system . . . it first arrives with the morning's mail . . . we're getting about 1000 pieces of mail on an average day. This is sorted out in the mail room and delivered to the various departments. Articles go first to *73's* Executive Editor, John Molnar, for a preliminary reading and some comments. Then they are read by me and I comment. From there they are sent to John Craig out in California for his decision and preliminary editing.

Rejected articles are returned by John and the accepted ones are sent back to New Hampshire, together with a request for payment. These go to Managing Editor, Kurt Schmidt, who sends the payment requests to Knud Keller (KV4GG), the bookkeeper, Kurt also makes a file card for each article for quick reference. Articles are then checked for spelling, grammar, and style of special symbols.

Any schematics or drawings are sent out for professional drafting. The copy is sent to the typesetting department to be set by Barbi Latti and Sandy White. If any photographs are needed which have to be taken by us we get Stan Miastkowski (WA1UMV) from our book department to take 'em with our Mamya RB67 camera.

When the drafting and typesetting are completed, the type is proofread by Peggy Sysyn, and Jody Wright then does a rough layout and paste-up of the article. A copy of this is sent to the author for his proofing.

Once the author's proof has been returned the article goes to the art department where Lynn Fraser assigns it to one of the seven artists there for final paste-up. This is a painstaking job, for it must be done precisely. A block of red plastic has to be cut the exact size of any photograph to be put in the article layout because photos have to be made separately and pasted on the page negatives later.

When the page is completed, it goes back to Kurt for a final inspection to make sure all corrections from proofreaders have been made. Then it goes back to the camera department where Bill Heydolph makes a negative of it on the huge copy camera. Separate negatives are made of any photographs which will be pasted in.

Noel Self (WB1ARP) or Robert Drew might glue the halftone photographs on the full page negatives. They would then check the page negative for any glitches and would "spot" them with an opague paint. Now the page negative is almost ready to be used.

After Bill Edwards, the ad manager, and Leslie Bailey, the advertising assistant, have firmed up the ads for an issue of the magazine they make a list of the ads to be used by the publisher when the page numbers are assigned to articles and ads. Kurt is also in on this "dummying" procedure when the dummy of the issue is put together.

The art department then has to make a negative of all of the page numbers, cut them into individual page numbers and insert one in each of the article pages. This is a delicate stripping job. Now the page negative has been completed and all that remains is for Bill Heydolph or Tedd Cluff to make a duplicate negative to be sent to the printer. The original negative has the stripped-in page numbers and perhaps some stripped-in ads . . . plus any pasted-on halftones. These might come apart when made into 32 page sections at the printer so we have to send duplicates which are all in one piece to prevent this disaster.

Once all the pages of the magazine are in negative form the whole works is packed in a box and sent to the printer in Connecticut for printing and mailing.

Handling subscriptions is a whole other big deal, with three girls processing subscription orders, three more entering them in the computer system, a computer staff of two to prepare invoices for subs to be billed . . . another girl to type in the names and addresses of prospective subscribers (sent to us by manufacturers) . . . a marketing department of three people to solicit newsstand wholesaler sales, sales of computer and radio stores, and direct mail subscription

continued on page 128

kilobaud T.M.

PUBLISHER
Wayne Green
EDITOR
John Craig
MANAGING EDITOR
Kurt Schmidt
ASSISTANT EDITOR
Jody Wright
EDITORIAL ASSISTANT
Peggy Sysyn
PRODUCTION DEPARTMENT
Manager:
Lynn Panciera-Fraser
Staff:
Craig Brown
Gayle Cabana
Robert Drew
Michael Murphy
Bob Sawyer
Noel R. Self
Robin M. Sloan
TYPESETTING
Barbara J. Latti
Sandy White
Marie Walz
PHOTOGRAPHY
Bill Heydolph
Tedd Cluff
DRAFTING
Bill Morello
Lynn Malo
ASSOCIATE EDITORS
Don Alexander
Sheila Clarke
Rich Force
John Molnar
MANAGER
Biff Mahoney
COMPTROLLER
Knud E. M. Keller
ASSISTANT COMPTROLLERS
Marge Nielsen
Robin Hunter
MARKETING
Sherry Smythe
Karen McDonough
Lisa Joseph
ADVERTISING
Bill Edwards
Leslie Bailey
Nancy Cluff
Barbara Hann
Janet Ames
Lisa Healey
CIRCULATION
Dorothy Gibson
Nancy Chandler
Janette Dyer
Florence Goldman
Judy Main
Jeane Shattuck
Theresa Toussaint
PURCHASING
Susan Brumaghim
COMPUTER DATA CONTROL
Judy Waterman
Judy Brumaghim
Sherry Dean
Mary Jo Sponseller
COMPUTER ENGINEERING
C. Robert Leach
David E. Wilensky
Richard Dykema
PRINTING
Michael Potter
John Bianchi
William Cering
Brent Lawler
Gary Steinbach
INVENTORY CONTROL
Marshall Raymond
Larry Ames
Gary Slamin
PLANT MAINTENANCE
Bill Barry
Lorraine Pickering

Kilobaud is published monthly by 1001001, Inc., Peterborough NH 03458. Subscription rates in the U.S. and Canada are $15 for one year and $36 for three years. Outside the U.S. and Canada, please write for foreign rates. Application to mail at second class postage rate pending at Peterborough NH 03458 and additional mailing offices. Phone: 603-924-3873.

20 Print "Editor's Remarks"
30 End
Run

John Craig

COMPUTER CLUBS

How important is your local computer club to you? What do you get out of those club meetings? More important, what do you contribute to those meetings? Do you feel your club could be doing some worthwhile things it isn't? These are just some of the many questions I've come up with lately since getting involved in some changes with our local computer club. Our club has always been a prime example of a nonclub. We simply have an informal meeting in which everyone introduces themselves and discusses for a moment what they're doing with their system and/or any information they would like to share with the group. Afterwards, we break up into a random access period in which people stand around in groups discussing things of mutual interest or looking over the systems we always have on display. The underlying qualities of this group are really important, I feel. Nobody takes the initiative and has to round up people to bring their systems for display. People just do it. As a matter of fact, we don't really have anybody doing much of anything. It's just a monthly meeting of about 40 people who get together for a few hours and discuss their mutual interests.

Recently, I asked a newcomer to our group if I would be seeing him at the next meeting. I was a little surprised at his answer. He said that he had better things to do than just sit around and chat and if he was going to get involved in a club it would be with one that was doing things! Well, this started me thinking that maybe we had been stumbling along in ignorant bliss and really should be doing some worthwhile things. So, I made up a questionnaire and passed it around at the next meeting. (This also gave me a chance to make an effort toward getting the name of the club changed, since I never have cared too much for *The Central California Computer User's Group.*) The questionnaire served two purposes: number one was that we were able to take a survey (for the first time) of all the people who owned systems, what kind they were, and what they were using them for. The second objective was, of course, to find out if there were a number of other people who felt the club should be doing more than it was, or if things should be done differently in order to make the get-togethers more meaningful. The results were interesting, but before I get into that, I want to discuss the first part of the questionnaire for a moment. If your club hasn't taken a survey of each member's equipment and applications and passed the results around to everyone, it seems to me it would be a very worthwhile effort to consider. I'm sure you've all noticed how helpful fellow computerists are. I've never seen anything to compare with it. But, if you've got a particular piece of equipment and a problem crops up with it, how are you going to know about all the other people in your group who also have that equipment? Taking that survey and distributing the results will be very helpful in this area (to say the least). And, if it keeps just one person from becoming discouraged and putting that home system up on the shelf, then it was certainly worthwhile.

We had 35 people attending the meeting the night I handed out those questionnaires and I got 35 questionnaires back. Having it mailed out in a newsletter or handing them out at one meeting and expecting people to bring them to the next one just doesn't work as well as taking a few short minutes and having everyone fill it out right there. An example of the type of questions I asked were, "Do you think the club should have dues?", "Should there be club projects and/or workshops?", "Should we have speakers (either members or manufacturers or both) occasionally or all the time?" and "Should the meetings be held once or twice a month?"

One of the most significant results was there was an overwhelming majority who wanted speakers (both manufacturers and club members). I think it was also significant that nobody wanted speakers at *each and every meeting.* We've never had speakers . . . but I guess we will from now on. You know, I've been around to quite a few clubs and I've seen very few really good speakers get up in front of a group. I've seen a lot of bad speakers who could have been much better. Whether it's a club member or a manufacturer's representative giving a talk to a group, I feel quite strongly that the person should submit an outline of his talk beforehand so that members of the club can get back to him with suggestions. This isn't really such a big deal. Nobody (unless they are very accomplished at public speaking) should get up in front of a group and give their spiel right off the cuff. To begin with, there should be objectives set down (i.e., just what is it that he wants to impart to the group?) and by following an outline the chances of getting sidetracked and not meeting those objectives are lessened. A time limit is of prime importance! How many times have you sat and suffered while the guy giving a talk has run over his allotted time . . . and he keeps going, and going, and going! (And how certain you are at the end that the applause is really to thank him for finally sitting down and shutting up!) The regular members of the club will suffer through a bad speaker now and then but I think one of the prime reasons for doing everything possible to eliminate this is so that newcomers won't be turned off.

One final point regarding speakers is that I feel quite strongly their presentation should be directed at the computer hobbyist. Having a minicomputer manufacturer or (even worse) a large-frame manufacturer come in and talk about his equipment doesn't strike me as being appropriate. On that note, it should be pointed out that the club should do a good job of providing the prospective speaker with a good description of his audience. Actually, a good speaker will ask.

I've got several other things I'd like to discuss regarding clubs (such as the role of the president, newsletters, club projects, getting new members interested, the random access periods, workshops, community activities and more). As you can see, I wouldn't have any trouble going on, and on, and on . . . and if there were any applause (hah!) at the end, it would surely be because I finally shut up!

I'd like very much to hear from you on how your club is run, what it's doing, and anything unique that you feel should be shared with the rest of the hobbyist community. Particularly, I'd like very much to see the results of surveys similar to the one I've mentioned here.

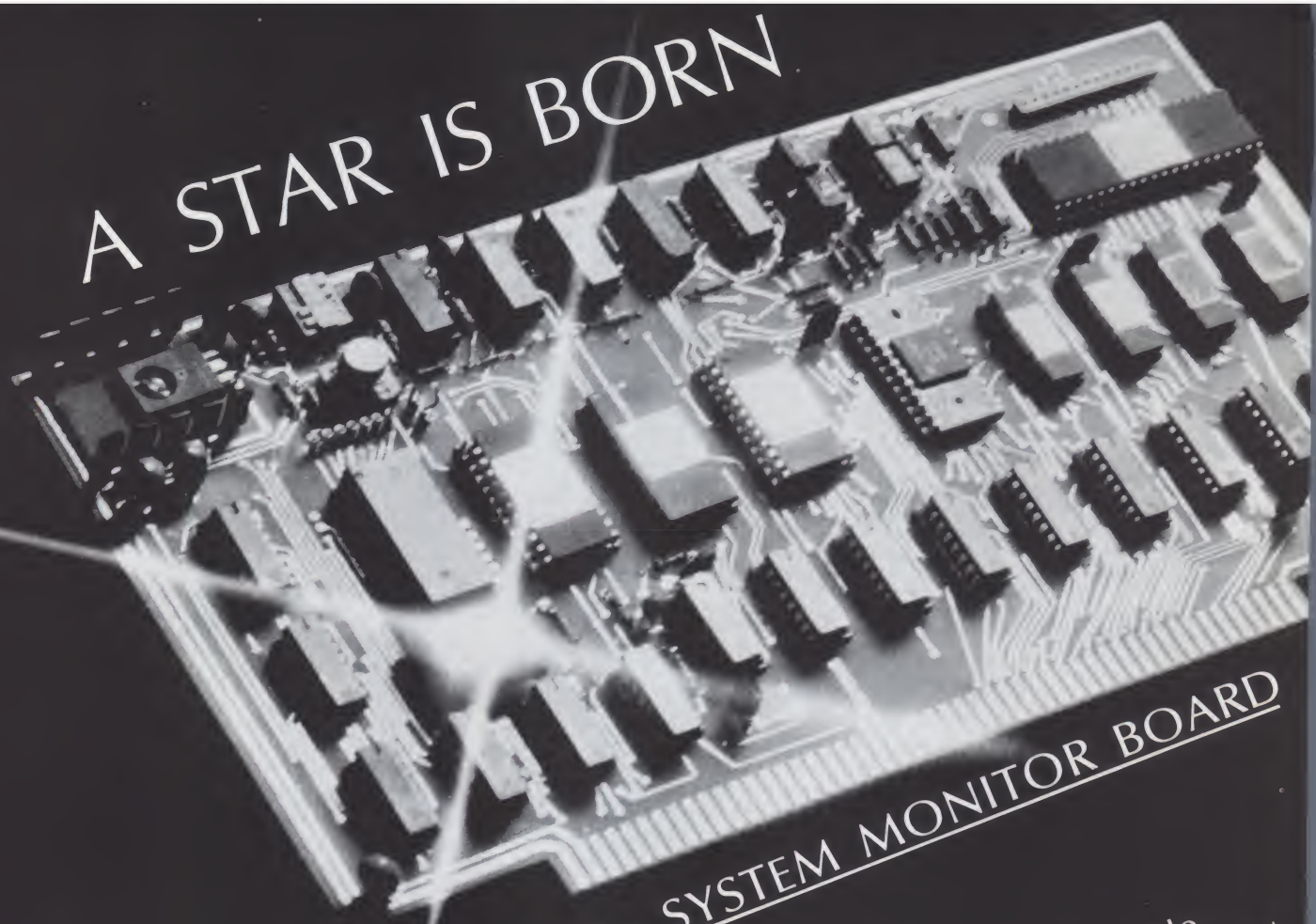
By the way, we're going to have an article in next month's issue by Charles Floto with some ideas on how to promote your club at (Are you ready for this?) the *county fair.* Why not?

THE "KILOBAUD GANG"

We've had several letters from readers which have been addressed to "The *Kilobaud* Group," "The *Kilobaud* Crew," and others. I like that. Quite often (especially with those letters of praise), the letter *should* be addressed to the crew. Take a look at the masthead on page three (You knew that was called the *masthead*, didn't you?). The years of experience in putting together magazines which can be found among those names is staggering. Many of those dedicated folks have been working for *73 Magazine* for many years and are now carrying double duty with *Kilobaud.* If you have a copy of the first issue of *Byte* laying around you'll be amazed at the number of people in the

continued on page 105

A STAR IS BORN



TDL's STARTLING NEW Z80 SYSTEM MONITOR BOARD

Replaces 4 basic boards in your system.
Integrates their functions fully into a single module.
Radically simplifies the development of your system.
Adds more versatility and performance than you'd believe possible.

- **Fully S100 bus compatible** • Power on start-up so no front panel is needed • Wait state circuitry • Fully solder masked and silk-screened Documentation includes complete software listings of the Zapple Monitor • Minimizes interfacing problems • Like having your own resident technical expert to help set up your system • Saves space for more cards • To duplicate the SMB, you'd have to spend $500 or more on separate boards.
- **2 Serial I/O and 1 Parallel I/O port.** Both serial I/O ports may be individually configured for any baud rate between 110 and 9600, in either RS232 or Current Loop configuration. The parallel port is 8 bits in or out with a status bit. All ports directly controllable by the monitor.
- **1200 BAUD audio cassette interface.** Has its own dedicated serial port and utilizes an asynchronous phase-encoding technique for high reliability and broad speed tolerance. Most important, ALL TDL software will be available on cassette in this format.
- **2K of high speed low power RAM.** Uses 1Kx4, 350ns static RAM chips, and does double duty for either add-on monitor routines (for your own special I/O) or program workspace.
- **2K Zapple Monitor in Masked ROM** gives you perfect system control direct from the console. It has 27 Commands (including 3 user defined) and offers program debug routines.

Let TDL's System Monitor Board star in your computer system. Hurry, use coupon below to order, or call (609) 921-0321

TECHNICAL DESIGN LABS

SYSTEM MONITOR BOARD

TECHNICAL DESIGN LABS RESEARCH PARK Building H 1101 State Road Princeton N.J. 08540

Kit $295 Assembled & Tested $395

Please send your SYSTEM MONITOR BOARD.
☐ **KIT ONLY @ $295.** ☐ **ASSEMBLED & TESTED @ $395.** **(N.J. residents add 5% sales tax)**

Name _____ **Street** _____ **Zip** _____

City _____ **State** _____

I enclose ☐ **Check** ☐ **Money Order** **Charge Card Data:**
☐ **BankAmericard** **Exp. Date** _____
☐ **Master Charge** **Exp. Date** _____

Signature _____

☐ **Send COD. I enclose 25% deposit**
☐ **Send your FREE CATALOG.**

TECHNICAL DESIGN LABS
Research Park Bldg. H
1101 State Road
Princeton, N.J.
08540

T-8

BOOKS

Microcomputer Dictionary and Guide
Charles J. Sippl
Matrix Publications, 1976, $15.95

At first glance it would appear that this large volume (over 650 pages) is the answer to any microcomputer user's dreams. Surely a book this big must contain any definition that could possibly be required. Wrong. Not only does it omit and inadequately define terms that are clearly common to microcomputer usage, it is absolutely stuffed with terms that have absolutely nothing to do with computers!

The book is divided into the dictionary proper and seven large appendixes. One first clue to an impending case of entry overkill is in the Table of Contents. All through the cover and author's preface the volume is referred to as a "microcomputer dictionary." However, when you arrive at the index, it has become "Definitions of Microelectronics Terms." Now there are many parts of the microelectronics field that have only marginal bearing on microcomputers. This injects large numbers of terms that are not needed and which make it harder to find those that are needed. In particular, the communications industry seems exceptionally well represented in this microcomputer book. For example, there are 23 definitions which pertain to the term "phase," of which maybe two have anything to do with microcomputers. There are many, many similar examples.

In addition to large quantities of superfluous definitions, many of the definitions which are provided are obviously drawn directly from *specific products* rather than from generally applicable computer principles. Since many manufacturers tend to redefine terms to enhance the descriptions of their products, many of the definitions are incomplete, and some are quite misleading.

Another fact that contributes to the bulk of the book is that many definitions are repeated in different places with only trivial changes. For example, there are over 30 definitions of the term "register." The definitions, some differing by only a few words, were repeated under titles like: "microprocessor registers," "MPU registers," "registers, microprocessor," "registers, data (microcomputers)," and so on.

One closing objection (there is only so much room in this magazine) is that the dictionary is virtually unillustrated. Many of the terms almost demand a figure of flowchart to show their operation, particularly to new users. Many of the proverbial thousands of words could have been saved and overall clarity much enhanced by use of some graphic aids.

This dictionary was probably rushed to market in order to cash in on the blossoming interest in microcomputers. That is not to be faulted. What is unacceptable is to call the book by one name when it is, in reality, something else. It appears that the book is a collection of definitions gleaned from dictionaries and user manuals from across the electronics field, all strung together with only minimal concern for relevance to microcomputers. Perhaps the name "Abridged All Electronics Dictionary" would be more accurate. At any rate, the book is not well suited for beginners or for a regular microcomputer user's quick reference. A book one quarter the size could easily convey 99.5% of all the useful information this book contains. ■

Dave Winthrop
Santa Maria CA 93454

Finite State Fantasies
Rich Didday
Matrix Publishers, 1976
$2.25, 8½"x11", 48 pages

Bizarre, educational, weird, humorous, strange graphic stories and drawings about computing. Obviously the product of a sick mind. Anybody seen any were-computers lately?

John Craig

Digital Troubleshooting
Richard Gasperini
Hayden Book Company, Inc.
50 Essex Street
Rochelle Park NJ 07662
180 pages, $9.94 (paperback)

This book is subtitled *Practical Digital Theory and Troubleshooting Tips.* This is a more descriptive than the ostensible title. Far more space is devoted to the basics of digital logic than to the theory and techniques of troubleshooting. The treatment of digital logic is at a very practical level with considerable attention to how things work, how they are packaged, nomenclature, symbology, etc. Troubleshooting is the primary topic only in three chapters, although various special tools and techniques are discussed briefly at other points in the text.

The first chapter is a brief introduction to the concept of digital (as opposed to analog) electronics and of *high* and *low* logic levels. The next chapter describes the various ways in which logic is implemented in integrated circuits (RTL, DTL, T^2L, etc.), the concepts of fan-in and fan-out, and the characteristics of common integrated circuit packages. Chapter Three introduces metal oxide semiconductors (MOS) and field-effect transistors (FET). Chapter Four describes basic gate configurations and the corresponding logic symbology.

Chapters Five and Six concentrate on troubleshooting. Particular emphasis is given to the use of the many new tools now available for use with digital integrated circuits; logic probes, pulsers, logic comparators, logic clips and the logic analyzer. These chapters also deal with the theory of fault isolation and with some extremely practical work bench techniques.

Following chapters deal with more complex components such as flip-flops, decoders, shift registers, displays and memory devices. The basic theory and operation of each device is presented along with some brief troubleshooting tips. The discussions are clear, well illustrated and easy to follow.

The last chapters alternate between hardware and theory. There are descriptions of how integrated circuits are manufactured and how to desolder an integrated circuit from a printed circuit board. There is a chapter on alternative systems of logic symbols, one on Boolean algebra, and one on where to obtain replacement parts. An appendix discusses manufacturers' numbering systems for integrated circuits.

In general the book gives a very clear and concise presentation of the basics of digital integrated circuits at a very practical hardware level. However, like all books, this one is intended for a specific audience. The reader must then decide whether or not he/she is part of that audience.

The book assumes at least a minimal knowledge of transistor circuitry, a reasonable assumption for a book nominally about digital troubleshooting. It also assumes in the discussions of troubleshooting that the reader is planning to repair existing equipment (something that was a working product and now is *not* working) rather than debugging a new and unproven design. The emphasis is almost entirely on finding and replacing a faulty integrated circuit, implying that the would-be troubleshooter already knows all about things like dirty contacts, loose solder connections and the other host of gremlins that can infect electronic devices. This would also explain the complete lack of discussion of using an oscilloscope. The author obviously assumes that the practicing troubleshooter has experience with and understands a scope.

In fact, this book is written for electronic repairmen to introduce them to digital electronic devices. It will also introduce them to the many new troubleshooting tools manufactured by Hewlett-Packard. This is probably not a coincidence, in view of the author's affiliation.

Digital Troubleshooting can be of considerable value to the computer hobbyist if he has some background in transistor electronics but is approaching either digital logic or integrated circuits for the first time. It will be of interest to the hobbyist who intends to do some troubleshooting or who merely wants a better understanding of his expensive hardware (and what that repairman is doing to it). The book is a readable and practical treatment of integrated circuit logic and that in itself is enough to recommend it.

A. H. McDonough
El Segundo CA

Computers in Society:
The Wheres, Whys, and Hows of Computer Use
Donald D. Spencer
Hayden Publishers
$5.50 Paperbound

Since this book was published in 1974, my first reaction was to wonder if it would be worthwhile reading. Because technology in the computer industry accelerates at an increasingly rapid rate, I discard once-useful books and magazines every few months. So, I surmise, publishers should remove obsolete publications from their rolls.

But flipping through the pages of this book gave me pause; not only does Don Spencer have a good track record (*Game Playing With Computers*, for one), but I, like many others, jumped into the computer revolution in midstream and the book appeared to offer a general background. So I read it, and didn't get far before realizing that I was enjoying it as much for Spencer's predictions as for his background information.

Computers In Society was written to provide us with the range of possiblities computers offer the public, covering fields such as medicine, law, engineering, transportation, business and education. Included is discussion or applications for the artist, writer, sportscaster, housewife, and single looking for a mate. In other words, there are very few of us whose lives escape affects of the computer revolution.

The first chapter gives us a general knowledge of the computer evolution. Spencer defends the electronic marvel against myths and errors, and describes its electronic and human-related limitations. He gives us a general coverage of the components which make up a typical system. Readers must keep in mind, however, that the advent of microprocessors hadn't occurred at the time of this book's writing, so one tends to find fault with some of his statements. To wit, "Input information is usually prepared by card keypunches or paper tape punching units. These units are sometimes called *data preparation units* and are never directly connected to the computer." In the same section, the discussion of software gave me a better understanding of how various types of programs and lan-

guages work. Winding up the first chapter, Spencer predicted, "The fourth generation of computers will probably appear during the late 1970s and will use very compact circuitry, thus further increasing the computational speed and reducing the cost of computers." One can't help but speculate on where in the scale of importance Spencer would place micros.

In his second chapter, "Computers in Society", much of the discussion deals with matters most of us are now familiar with. As he explores the moral and legal ramifications of the use of personal data generated by computers, we are now aware of the legislation since enacted to protect individuals against invasion of privacy, and more laws being generated as the computer continues to threaten invasion. His proposals and predictions for a checkless, cashless society are now far closer to reality in 1977 than imagined in 1974.

A great deal of space is devoted to computers in medicine. Spencer must have spent enormous research in this field because his discussion comes off with a high degree of credibility compared to current reports of computers in medicine. Those in medically related fields might now find this section valuable as a reference for setting up diagnostic systems, monitoring patients, handling billing and drug inventories, and even directing appointments.

Twenty-two pages of Chapter 4 are about computers in fine arts, but I felt the subject was only lightly touched on. Although we get a look at many possibilities afforded by computers in writing music, animation, poetry and literature, Spencer does not mention that there are musicians' organizations whose main interest is in computer-generated music. Or, that almost all videotaped shows on television have been edited by, and partially produced via computers. However, the technique is briefly discussed using a single show's example. That might indicate that too much is going on for one man to know . . . or to include in a single chapter. There are, nonetheless, some extremely interesting processes which he describes, such as 12-foot pictures of a nude, a telephone, and others using a technique developed by Bell Telephone Labs. A number of photos in this chapter illustrate the diverse possibilities for graphics alone.

"Computers and the Law," Chapter 5, talks about the growing implementation of computers for information and communications systems . . . something most of us are now familiar with.

The chapter on "Computers in Engineering" might well have been placed ahead in the book toward the beginning. Especially since, as Spencer points up, "The engineer has been associated with the development and usage of digital computing equipment from the beginning. In fact, the engineer created much of the original demand for computers out of his need to solve problems encountered in military applications." We read on to become aware that other technological advances we take for granted would not have been possible without computer assistance such as space travel, satellite weather reporting, and architectural and automotive design.

Matters covered in "Computers in Business" and "Computers in Education" are subjects most of us now accept as a part of our daily lives. When Spencer predicted that ". . . in years to come computerized supermarkets may become rather commonplace," he may not have known that now, at least in Southern California and major cities throughout the U.S., his prediction has become reality. Descriptions of how those systems work gave me insight into an operation that, until now, I'd taken for granted and given little thought to when being checked through with my groceries.

The book continues describing the use of computers in defense, taxation, machine tooling and control, and farming. More interesting to the hobbyist might be the background of the advent of game playing, beginning with the first automatic chess-playing machine in the eighteenth century.

The book finishes with a discussion of computers in the future as the author sees it. Included is a brief mention of minicomputers (most of the book deals only with large scale systems). Absolutely no reference to the possibility of micros was made. I must assume that the possibility was nowhere in Spencer's imagination then. Even he did not guess that less than a year after publishing his book, the first microcomputer system would initiate itself into hobbyists' homes.

Despite these criticisms, it was fun to read predictions, noting how very close this author came to being accurate. Some of Spencer's statements included, "Within the next few years integrated-circuit memories are expected to be common . . . The primary characteristic of tomorrow's computers will thus be much the same as today's: an improved price/performance ratio." And, "Future computers will probably make extensive use of *firmware* . . ." And, "Cathode-ray-tube displays will be much more common in future computer applications, and the cost of these devices should decrease considerably." However, when he gets to "Computers in the Home", he missed the boat. Yes, they made it into our homes, but not as Spencer foresaw. (Though he stated the year 2001 for the time when we would all have a console with which to plan menus, shop, calculate checkbooks, etc. . . . and he's probably right.) But he skipped the hobbyist movement altogether . . . and that computers are *fun*. I suspect that we cannot accuse Mr. Spencer of having a limited imagination; he has covered the use of computers and has bravely made predictions that only a great deal of research and insight could have provided. Perhaps then, none of us can predict the all-encompassing range of possibilities that computers hold for the future, except that they will most certainly influence everyone in developed nations, and more directly than we've known in the recent past.

An excellent prose glossary follows the text, written using terms in context, and followed by an index of terms in alphabetic order. This organization would tend to make an easy reference for the novice.

Although each chapter is concluded with a list of reading references, I'm skeptical that the recommended books would be of value now, since their dates range from 1965 to 1974.

Computers In Society though still valid despite its 3 years of age, has become archaic. If Mr. Spencer has considered updating the book, a second edition which includes the coming of microprocessors, microcomputers, and hobbyists would definitely be of value, both at home, in the office, and in the classroom.

Sheila Clarke
Glendale CA 91206

LOOKAHEAD
HOME COMPUTERS
HOT AND COOL

Rich Diddday

Say – do you happen to remember Marshall McLuhan? Back in the sixties he wrote a couple of books[1] that caused a bit of a stir[2]. For a while there, the air was full of quotable quotes and strange phrases ("global village," "the medium is the message, massage ... whatever," "Now, bubble gum wrappers. Are bubble gum wrappers a hot or a cool medium?"). The fancy phrases are gone, but (for me, at least) the content lingers on. I thought it might be fun to take some of the ideas implicit in McLuhan's work and see what sorts of predictions they lead us to when we apply them to home computers.

First, let me lay out three main ideas of McLuhan's (as I understand them); then let's see how they apply to familiar situation in which a large fraction of the society has some form of home computer.

Principle 1: *Many of the major effects of a communications medium can be detected by ignoring the content of the medium and concentrating on what people do and do not do when they use it.* Stated another way, this is: assume that the content of the medium is sufficient to hold the interest of the user, then ignore the content and figure out what the user's body is doing. Does the user have to stay in one place? Can the user do anything else while using the medium? What kind of equipment is required? Does the user consume anything in the process (paper, electricity, water, food, etc.)? Are any particular sets of muscles used extensively?

Principle 2: *There are definite limits to the rate at which humans can process information, and we deal with media which tax our information processing capabilities differently from the way we deal with slower paced media.* McLuhan called high rate media "hot," low rate media "cool," and argued that "hot" media force the user into a "specialist" mode – to concentrate on subparts of the presentation, to adopt a "point of view," to remain somewhat aloof so as to be able to make the decisions about what to attend to, what to ignore. On the other hand, low rate "cool" media encourage the user to grasp the presentation as a whole, to "go with the flow," to be more of a participant, to have a deep emotional involvement with the subject matter.

Principle 3: *Different forms of media establish different perceptual and behavioral habits in people. Since a society is characterized by the intercommunications of its members, societies will be altered by the acceptance of new forms of media.* So, McLuhan claimed that societies in which hot media predominate tend to consist of specialists and to be fragmented, dispassionate, and "rational"; societies in which cool media predominate have more generalists, are more emotional, "mystical," and village-like.

Principles 1 and 3 seem pretty reasonable. There are some problems with the way McLuhan applied Principle 2, mainly because nobody really understands the psychological principles of how we process complex multisensory inputs. The best way to think about it seems to be this: the more attention the user has to pay to get all the details, the hotter the medium; the more decisions the user has to make, the hotter the medium. It is clear from Principle 2 that movies seen in theaters are "hotter" than the same movies seen on TV, or that the sound in a discotheque is "hotter" than the sound signals from a car 8-track recorder. But comparing the effects of purely auditory media like 8-track recorders to mixed visual/auditory media like TV gets pretty confusing (to me at least). We'll see what happens.

This might be sounding a little weird so far if it is all new to you, so let's go over an example before plunging off into the wild blue yonder. At this very moment, you are interacting with a hot, predominately visual, medium. Does it seem strange to think of the printed page as a *visual* medium? Maybe it seems as if I'm *talking* to you, but of course, I'm not; I've sent you a message in a form that you decode using your eyes. Does it seem strange to call the printed page a *hot* medium? Notice how incredibly specialized your eye movements are as you decode this. There's so much information that you can't grasp it all at once like you can a cartoon, you have to focus on a tiny portion of the page at a time, in a specialized order, to get the meaning. Compare the way you look at and feel about a cartoon ("cool") with the way you have to process the same sized area that's filled with printed matter ("hot").

I said that *Kilobaud* is a hot, *predominately* visual medium. Why *predominately?* Well, feel the paper. It's smooth, right? *Kilobaud* is communicating the fact that it is a classy magazine to you through your sense of touch.

Using Principle 1, I can fairly safely say that at this moment, you are not skiing, driving a car, or jogging. You are probably sitting or lying down in a place where there is a reasonable amount of light and heat, not much wind, etc., etc.

Here's another example. Movies seen in theaters are hot, TV pictures are cool. To see that just consider how many bytes of memory you'd need to display a typical movie scene versus a typical TV screen-full. Not only is the movie scene much larger, it is also of much finer resolution. Because of this difference, showing a movie made for one medium on the other changes the effect it has. If you saw the movie *All the President's Men*, you may recall how overwhelmingly forceful the opening scenes of President Nixon's helicopter were. They were scenes shot with a TV camera, blown up and "heated up" to movie size and fidelity.

Now let's take the three principles, adapt them to situations involving home computers, and see what we get. To apply Principle 1, let's assume that at some point in the future, home computers have evolved to the point where millions of people accept them as useful and fun enhancements of their TVs. If a show on network TV is boring, they can get out their keyboard and joystick and draw beards on the faces on the screen, or they can select any of a huge number of games, interactive story-telling programs, cartoon-drawing programs, accounting programs, etc. How will what they do differ from what they do with a purely passive TV? Well, if they use their TV in an interactive mode, quite a few things change. You can't iron shirts and play Star Trek at the same time. You can't play solitaire, work crossword puzzles, knit scarves, or discipline the kids while you're involved with the system. And on Sunday afternoon, you can't drink very much beer and still be able to type accurately! So, if you believe this analysis, in a decade or so, you would, I presume, sell your Parker Bros. and Budweiser stock and invest in companies that make pillows and specialized chairs for interactive TV addicts. Probably the audience size for network TV will go down, and possibly more people will listen to radio ("And now another hour of music to play Star Trek by . . .").

How does Principle 2 apply to home computers? Since it is hard to guess at the resolution of systems in the future, maybe we should start closer to home. It is not hard to think of situations that involve a cool interaction – take the game of Star Trek as it is usually played today. The graphic display on the standard form of Star Trek is extremely low in resolution even compared to TV, and it is updated much more slowly. Instead of seeing a Klingon warship in all its angular, evil glory, you see +++. Instead of the roller rink sized bridge in the Enterprise, you see < * > . The graphic scene in Star Trek then is extremely cool, and as predicted, the users have an emotional response in depth. Instead of having to devote effort to understanding the scene, you are occupied with your private fantasies about what's going on. Now, some forms of the game are heated up by adding a huge number of options for the player to choose from and by lots of (hot) text for the user to decipher. This raises an interesting question which we could put to experimental test – would heating up the game through some combination of increased resolution of the graphic display and increasing numbers of options for the user to choose among *destroy* the intense sense of participation and involvement that devoted Star Trek players seem to get? At what point will the game shift from its current "mystical," cool, emotional form to a rational, more chess-like, intellectual game? People who are trying to sell game programs would do well to investigate this – it seems likely that by naively adding more options to make a "Super Star Trek," you might take away the very things that make it so popular.

There are lots of examples of using home computers in hot interactions. Programming in raw machine language is an extremely hot, specialized activity. The higher level languages are cooler than machine language but still very hot. Programming may be so hot that it will be a specialization practiced in detail by a minority of future home computer users.

Principle 3 says that using certain types of media creates ingrained habits in people dealing with their day-to-day lives and that knowledge of the dominant medium in a culture gives strong clues to the nature of interactions in that culture. Let's again make the assumption that some kind of "interactive TV" with a wide range of available programs is adopted by millions of people. The question is, will such a medium have the effect of fragmenting the society (as print media do), or will it increase the trend toward emotional involvement (as TV does)?

Basically, the medium I've hypothesized represents our current TV medium with the addition of the ability of the user to play an active

continued on page 21

POPULARITY EXPLOSION!

JUPITER II A
6800 System
$795

ASSEMBLED

JUPITER III A
Z80 System
$865

ASSEMBLED

If you thought the quality of a wire-wrapped system was beyond your price range — Take a look at what we have now!

The Jupiter IIA and the Jupiter IIIA Basic computer systems. You get the system module cage with fully assembled backplane, fully assembled plug-in ferro-resonant power supply, front panel and your choice of 6800 or Z80 CPU module. All less than the price of the two best selling 8080 systems!

Plus you can choose from the fastest growing selection of memories and peripherals available from any manufacturer, like our 2KB EPROM/ 4KB RAM/serial RS-232 module and our new 1024 character video module. Both can transform your basic computer system into a real star.

And remember, all Wave Mate products meet the highest quality industrial standards, with rugged construction unmatched by anyone! Join the popularity explosion and get yours now! Write or call for more info and your closest Wave Mate authorized distributor.

You get your choice of microprocessors!
And you get wire-wrapped modules too!

Now you have a low cost way to get started into personal computing without sacrificing future growth capability!

Send information on: ☐ Jupiter IIA system
☐ Jupiter IIIA system

NAME _____

ADDRESS _____

CITY _____ STATE _____ ZIP _____

Wave Mate

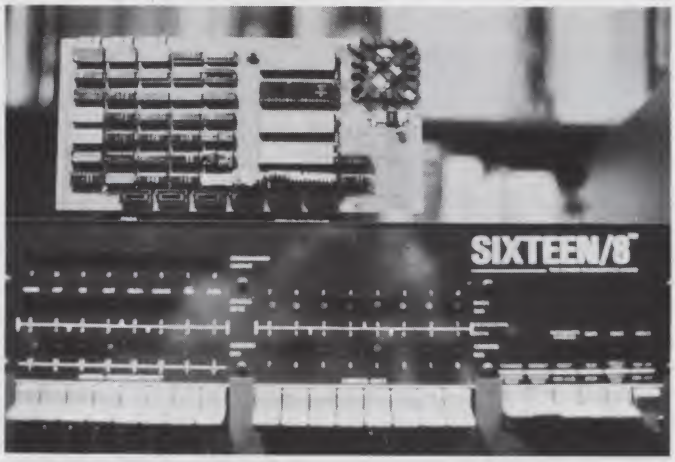
WAVE MATE 1015 West 190th Street, Gardena, California 90248 Telephone (213) 329-8941
Dept 24

W-5

Around the Industry

Sixteen bits on the Altair bus! My imagination was fired up when I first heard about Dick Wilcox's creation, but after having him demonstrate it, I feel like I've come down from some kind of an emotional high. And getting really excited about a computer demonstration isn't the easiest thing to do when you've seen dozens of them before. But . . . not only has he developed a rather dynamic system, Dick has a tendency to become rather dynamic when he's putting it through its paces. The most impressive part was when he showed me its multiprogram execution capabilities. He started off by having two, and then three programs running together (under interrupt control). Then, like a movie director leading up to a climatic moment, he kept adding more programs until he had *eight* running at the same time!! So that you could actually *see* this happening, he had a most impressive demonstration program! Most exciting watching his monitor screen with this nutty program running. I just wish we were dealing with moving pictures here instead of stills! There were two bright *lightning bolt* symbols gently flittering all over the screen, and when they touched the edge, they would cause one of several blocks to be created. The idea was for them to be *building* a city. A real megapolis! A closer examination showed six circular characters (two of them trapped in the boxes at the bottom). These little guys go flittering around just like the lightning bolts, except when they bump into one of the blocks that they were putting up, they tear it down! Fascinating! When, and if, some of you game phreaks out there see this particular demonstration, you will go totally out of your minds. Can you imagine having several games within the computer playing against each other? Or, how about two or three playing against you?

Now, all of that was fine and dandy, and made for a very impressive demonstration but let's look at the serious side for a moment. The program was simply a tool to demonstrate the interrupt handling capabilities of the machine and the sophisticated software Dick has developed to go with it. We haven't seen anything like it in personal systems and the reason is because nobody has developed a *time-sharing* system before. The eight programs he had running in that demo program could just as easily have been eight users, each with a terminal, running their programs *simultaneously*. (The programs aren't really running at the same time. Under interrupt control they all *appear* to be running at the same time.) Small business systems with several terminals for data entry and multistudent systems in schools are just a couple of examples of the value of a good time-share system (not to mention the multiusers of the home system in the years to come).



One of the two CM-16 boards shown on an extender above the Imsai cabinet in which it resides.

It was way back in September of last year at PC '76 in Atlantic City that I first heard of Dick's machine during a conversation with John French. John and Dick have since formed a company called Alpha-Micro Technology which will be the vehicle for manufacturing and marketing the machine. A third member of the team is John's daughter, Debra, who not only brings additional enthusiasm into the project but she's a heck of a lot better looking than John and Dick put together!

The system has two things going for it: the fact that it is designed around a 16-bit microprocessor and the sophistication and quantity of software which will be available. The processor is the Western Digital MCP-1600, which consists of two 40-pin system chips and from one to four microprogrammable ROMs. The ROMs are microprogrammed to produce a customized instruction set and Dick has dubbed his particular version the WD-16. The WD-16, along with about 70 TTL chips, is incorporated into two PC boards which plug into an Altair bus machine. The whole thing is called the CM-16. I'm not going to go into some of the super hardware features of the CM-16, or discuss the Disc Operating System, BASIC, text editor and other software goodies because Dick will be writing an article on the system for next month's issue. Well, what the heck, I really should tell you about *some* of those things! For example, the assembly-language development system is just that . . . a development system. Totally relocatable code for the CM-16 instruction set is possible . . . there's an 8080 cross assembler (and would you believe the 8080 CPU board can share the bus with the CM-16 . . . so after assembling the program, control can be turned over to the 8080 and it can then run it) . . . a disk operating system which features all (and a lot more) of the techniques Dick has mentioned in his series on developing a *home brew* operating system . . . and I'll leave the BASIC and text editor for him next month.

Demonstration of the Alpha-Micro Technology CM-16 for John Craig.

NO ONE PUTS THEM TOGETHER

LIKE WAVE MATE

Now Wave Mate puts them together for you — the Jupiter system, the new high performance dual floppy disk drive from PerSci and new flexible software.

Imagine what you can do with a disk drive that seeks over five times faster than the closest competitor.

Imagine what you can do with a computer system that's wire-wrapped so it can be upgraded with advancements in technology.

Imagine what you can do with a series of high level compilers so flexible that the software you write for today's hot microprocessor will run on tomorrow's.

No one but Wave Mate can put a flexible package like this together for you.
Can you imagine any reason why you should settle for less? We can! You can start smaller with the Jupiter A system without sacrificing the quality and future growth capability of your computer system and you have your choice of 6800 or Z80 processors.

Send information on:
- ☐ Floppy disk system
- ☐ Jupiter IIA system
- ☐ Jupiter IIIA system

NAME _____

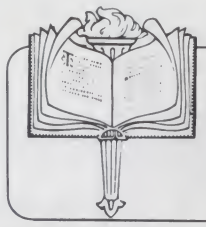
ADDRESS _____

CITY _____ STATE _____ ZIP _____

Wave Mate

WAVE MATE 1015 West 190th Street, Gardena, California 90248 Telephone (213) 329-8941
Dept. 22

W-5



The BASIC Forum

Dick Whipple/John Arnold

STRING MANIPULATIONS

In the last BASIC Forum, we discussed a technique for saving machine language programs using BASIC. In addition, it is possible to save the value of numeric and string variables in a BASIC program. With some BASIC interpreters no direct way is provided to store data entered by the user or generated by the BASIC program itself. In such cases the techniques described below provide a way to store data on a mass storage medium such as a cassette tape. Because the techniques involved differ slightly, we will consider numeric and string data separately.

A string is simply a list of alphabetic or numeric characters assigned to a certain variable name. Strings are of a specified length, usually less than 256 characters. Strings can be created by INPUT, LET, and READ statements. String variable names differ from numeric variables in that the last character of the name must be a "$". For instance, consider this statement:

```
LET A$ = "TEST STRING"
```

After execution, the characters within the quotation marks are stored in A$. The string can be recalled for later use by reference to the variable name A$.

BASIC interpreters generally provide certain functions for manipulating strings. For instance, in MITS 8K BASIC the following functions are available:

LEN (string expression) – Returns the number of characters in the string expression (see Example 1).

```
10 LET A$ = "TEST STRING"
20 PRINT A$,LEN(A$)
30 END
RUN
TEST STRING      11
```

Example 1.

STR$ (arithmetic expression) – The numeric value of the arithmetic expression is converted to a string. Numeric values are stored in binary form, not as the ASCII string that one sees when, for instance, the value is printed. It is sometimes advantageous to store the value as a string in which case the STR$ function is used. Consider the program example:

```
10 LET A = 6.48
20 LET B$ = STR$(A)
30 PRINT A,LEN(B$)
40 END
RUN
6.48             5
```

The program is used to determine how many characters are printed when A is output. Note, there is a leading space before the 6 which is counted as a character.

VAL (string expression) – Used to convert a string expression to a numeric value (just the opposite of STR$). The conversion proceeds from left to right in the string and is terminated by the first character other than (1) leading spaces (2) decimal point or (3) the numbers 0 to 9. The directly executed statements below illustrate the VAL function.

```
LET C$ = "22.6"
PRINT VAL(C$) + 6
28.6

LET D$ = "TEST 12"
PRINT VAL(D$)
0
```

CHR$ (arithmetic expression) – Produces a one character string whose character is the ASCII equivalent of the numeric value of the expression. Of course, the numeric value must be between 0 and 255 decimal. Consider this example and refer to Table 1 for the ASCII values.

```
10 LET X = 65
20 PRINT CHR$(X)
30 END
RUN
A
```

Referring to Table 1, note that the ASCII character whose decimal equivalent is 65 is the letter A.

ASC (string expression) – Converts the first character of the string to the ASCII numeric value. Again refer to Table 1 and the example below:

```
10 LET A$ = "APPLE"
20 PRINT ASC(A$)
30 END
RUN
65
```

MID$ (string expression, arithmetic expression #1, arithmetic expression #2) – Used to create a string by selecting characters within

Decimal Value	ASCII Character	Decimal Value	ASCII Character
0	NUL	64	@
1	SOH	65	A
2	STX	66	B
3	ETX	67	C
4	EOT	68	D
5	ENQ	69	E
6	ACK	70	F
7	BEL	71	G
8	BS	72	H
9	HT	73	I
10	LF	74	J
11	VT	75	K
12	FF	76	L
13	CR	77	M
14	SO	78	N
15	SI	79	O
16	DLE	80	P
17	DC1	81	Q
18	DC2	82	R
19	DC3	83	S
20	DC4	84	T
21	NAK	85	U
22	SYN	86	V
23	ETB	87	W
24	CAN	88	X
25	EM	89	Y
26	SUB	90	Z
27	ESC	91	[
28	FS	92	\
29	GS	93	]
30	RS	94	↑
31	US	95	←
32	SP	96	`
33	!	97	a
34	"	98	b
35	#	99	c
36	$	100	d
37	%	101	e
38	&	102	f
39	'	103	g
40	(	104	h
41	)	105	i
42	*	106	j
43	+	107	k
44	,	108	l
45	–	109	m
46	.	110	n
47	/	111	o
48	0	112	p
49	1	113	q
50	2	114	r
51	3	115	s
52	4	116	t
53	5	117	u
54	6	118	v
55	7	119	w
56	8	120	x
57	9	121	y
58	:	122	z
59	;	123	{
60	<	124	|
61	=	125	}
62	>	126	~
63	?	127	DEL

Table 1.

the original string expression. The substring begins at the position defined by the value of the expression #1 and continues for the number of characters specified by expression #2. Actually, expression #2 is optional. When omitted, the new string continues through the end of the original string (see Example 2).

```
10 LET A$ = "BEARCAT"
20 LET B$ = MID$(A$,1,4)
30 LET C$ = MID$(A$,5)
40 PRINT B$,C$
50 END
RUN
BEAR          CAT
```

Example 2.

With these string functions, it is possible to separate the characters of a string. If the characters are separated one at a time, and in order from left to right, we will call this "scanning the string". Fig. 1 is a representation of the string created by the BASIC statement below:

```
LET A$ = "TEST STRING"
```

The variable I will be used as a pointer to keep track of the character being separated from the string. Suppose we wanted to separate the first character (a "T"). Consider the short program shown in Example 3.

In the MID$ function of line 30, the second argument, I, indicates which character is to be separated and the third argument, 1, indicates that only a single character will be separated. Scanning of the string is achieved by placing line 30 within an appropriate FOR-NEXT Loop with I the index variable. The lower limit of the loop will be 1, while the upper limit will be set to the length of the string as determined by the LEN function. The program is shown in Example 4.

```
10 LET A$ = "TEST STRING"
20 FOR I = 1 to LEN(A$)
30 LET B$ = MID$(A$,I,1)
40 PRINT B$" ";
50 NEXT I
60 END
RUN
T E S T  S T R I N G
```

Example 4.

In order to store a string on tape, it must be scanned so that only one character is transmitted to the tape device at a time. The technique illustrated in the program in Example 4 becomes the heart of a string-saving program. Before presenting such a program, there is one additional point to be considered. The MID$ function is relatively slow in relation to most programming operations. As this can create timing problems, it is better to scan and store the ASCII value of each character as an array element first. The array elements can then be transmitted to the tape device at a much greater rate. (The MID$ function having been eliminated during the actual dumping operation.)

The program in Example 5 illustrates how a string (A$ in this case) is stored. Lines 200-210 are the tape output statements which may vary depending on your particular system. (For more discussion on this point, see our previous BASIC Forum.)

To read a tape generated by the program in Example 5, one merely reverses the procedure. The data which has been stored on the tape character by character is first read and placed in an array. The end of string mark (255 decimal) is used to terminate this portion of the program. The array elements are then converted back to single character strings which are concatenated (linked) to reproduce the original string. Example 6 is a program illustrating the procedure.

```
10 LET A$ = "TEST STRING"
20 LET I = 1:REM POINT TO FIRST CHARACTER
30 LET B$ = MID$(A$,I,1)
40 PRINT A$,B$
50 END
RUN
TEST STRING      T
```

Example 3.

```
 95 REM STRING SAVE PROGRAM
100 DIM T(50):REM DEPENDS ON SIZE OF STRING
110 REM SET STRING IN A$
120 LET A$ = "TEST STRING"
130 REM SCAN AND STORE IN ARRAY T
140 FOR I = 1 to LEN(A$)
150 REM ASC USED TO CONVERT TO NUMERIC VALUE
160 LET T(I) = ASC(MID$(A$,I,1))
170 NEXT I
175 LET T(I) = 255:REM END OF STRING MARK
180 REM TRANSMIT ARRAY T TO TAPE DEVICE
190 FOR I = 1 to LEN (A$) + 1
200 WAIT 6,128,128
210 OUT 7,T(I)
220 NEXT I
999 END
```

Example 5.

```
  5 REM STRING LOAD PROGRAM
 10 DIM T(50)
 20 LET I = 0
 30 LET I = I + 1
 40 WAIT 6,1,1
 50 LET T(I) = INP(7)
 60 IF T(I) <> 255 THEN 30
 70 LET J = I - 1
 80 LET A$ = "  "
 90 FOR I = 1 to J
100 LET A$ = A$ + CHR$(T(I))
110 NEXT I
120 PRINT A$
130 END
```

Example 6.

Caution! The maximum data rate depends on the execution time of the BASIC

continued on page 128

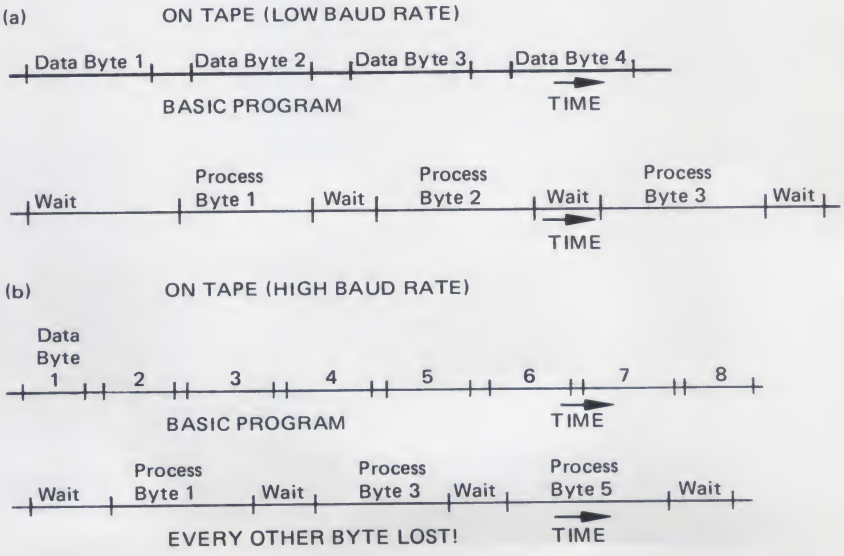


Fig. 2. The two timing diagrams illustrate the problem encountered when BASIC processing takes too much time. Case (a) with a low baud rate, all data correctly processed. Case (b) with a high baud rate, some data is passed over due to processing lag.

NEWS OF THE INDUSTRY

6800 AND 8080 COMPATIBLE LOW PRINTER – MODEL IMP-1

Electronic Product Associates, Inc., has announced the availability of a new, low-cost, 40 column, dot-matrix impact printer. The printer complete with drive electronics, character decoding and software driver PROMs, power supply and attractive hardwood and plastic cabinet interfaces directly with the 6800 and 8080 microprocessors. The printer is capable of printing a surprising 80 character per second bi-directionally. Single quantity pricing is $450.00, delivered from stock.

The IMP-1 utilizes a serially-driven printing element consisting of 7 print solenoids and print wires. The print wires are arranged vertically; the printing element is driven from either direction at constant speed. A synchronous motor driving a spirally grooved drum accomplishes this motion.

Ribbon feed is accomplished as a simple by-product of printing element motion. Ribbons are inexpensive and easily replaced.

All electronics for driving, decoding and program storage are powered by the self-contained dc power supply.

For further information contact: Electronic Produce Associates, Inc., 1157 Vega Street, San Diego CA 92110, 714-276-8911.

Electronic Product Associate's IMP-1 printer.

THE ELECTRONICS SOURCEBOOK

How to obtain 101 FREE samples, handbooks, catalogs, manuals and applications notes. Are you an electronics hobbyist, experimenter, ham or CBer who enjoys building electronic projects? If so, how would you like to obtain free parts and publications to help you in your projects? If you can answer YES to either of these questions, you'll be interested in an exciting, new publication called the *Electronics Sourcebook*. It contains first-hand information on some exicting news in the electronics industry that is of interest to you.

There are seven chapters:

The Electronics Bonanza
– Useful Background Information
– What This Book Can Do For You
– What Types of Items Are Available
Where To Find Unlimited Sources Of Free Samples & Publications
– The key to finding Unlimited Sources of Information
– Two Directories with over 6000 Pages of Useful Information
– Ten Important Reference Books
– 13 Informative Trade Publications
How To Obtain The Items You Want
– Four Rules That Insure Success
– How To Obtain Free Samples
– How To Obtain Free Publications
Communications
– Free Handbooks, Manuals, and Catalogs Relating To Ham, CB & SWL Equipment
Microcomputers
– Free Handbooks, Manuals & Application Notes Relating To Microprocessors, Microcomputers and Peripheral Equipment.
Electronic Components
– Free Handbooks, Samples, Manuals, and Application Notes on Transistors, ICs and other Components
General Electronics
– Free Literature and Catalogs Pertaining To Books, Test Equipment, Audio, Tools & Much More

Available from Technical Publications, 1405 Richland Ave., Metaire LA 70001, for $3.50 each, plus 25¢ postage.

THE DESIGN MATE-4

The Design Mate-4, a highly versatile, laboratory quality pulse generator serves a wide variety of industrial and institutional digital applications, yet lists for a low $124.95.

The DM-4 is designed to fill the bill wherever a source of clean, crisp, fast output pulses compatible with virtually all logic families and discrete circuits is required. It is capable of generating symmetrical and unsymmetrical pulses from 0.5Hz-5MHz and has a positive output of 100mV to 10V, with a rise and fall time of less than 30 nanoseconds. Additionally, DM-4 offers an independently controlled pulse width and spacing from 100 nanoseconds to 1 second in 7 overlapping ranges, as well as independent variable amplitude CMOS, and fixed amplitude TTL outputs. An independent TTL compatible sync pulse leading the main outputs by 40 nanoseconds is also provided. Design Mate-4 may be used as a clock source, delayed pulse generator, synchronous clock source, manual system stepper, pulse stretcher, clock burst generator and in tandem with one or more DM-4s used to gate the output of one or more additional DM-4s. Its duty cycle range is 10-7:1 and the unit operated either continuously or in manual one-shot fashion. It also features external triggering from dc to 10MHz and synchronous output gating.

Because of its great flexibility and low price, the DM-4 is designed to appeal to engineers, scientists, technicians and students for use in research and development, quality control, production testing, maintenance and troubleshooting throughout the electronics industry, as well as in the growing number of other fields which have *gone electronic* either in their products or production facilities.

For more information, contact Continental Specialties Corporation, 44 Kendall Street, Box 1942, New Haven CT 06509.

88-ANALOG/DIGITAL CONVERTER

The newest addition to the MITS Altair 8800 system is the 88-Analog/Digital Converter: a 12-bit card which permits the Altair to measure analog voltages often encountered in scientific and industrial applications with an accuracy of one part in 4096.

The 88-ADC is completely bus-compatible with the Altair 8800a or 8800b and is easily accessed using 8K BASIC.

The heart of the 88-ADC is the analog-to-digital converter module which contains virtually all of the circuitry needed to represent the analog voltage as a 12-bit binary value.

The new 88-ADC also includes a buffer amplifier (with a true differential input instrumentation amplifier option), an 8-channel multiplexer

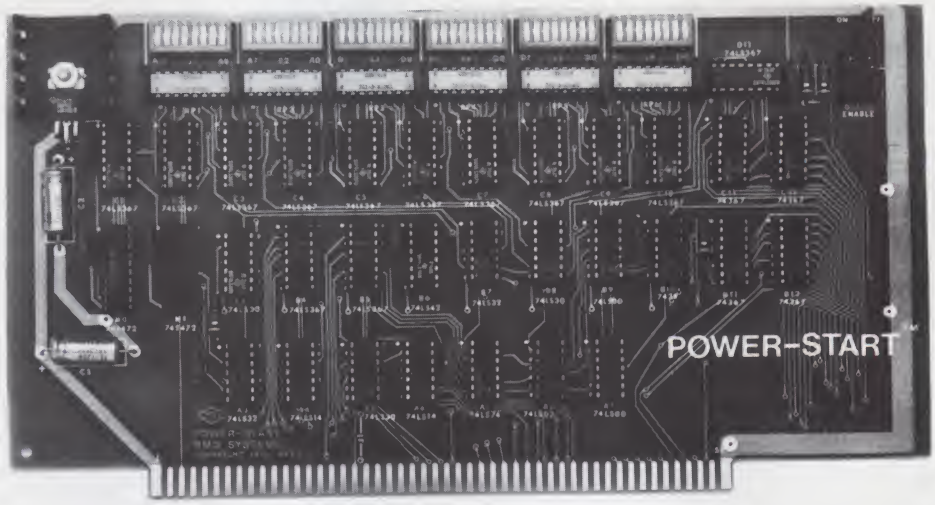
The Design Mate-4 from Continental Specialties.

(used to select one of the eight input signals), circuitry to address the card (the ADC is treated as an I/O device) and the associated timing circuitry.

A 24-channel multiplexer (88-Mux) card is available as an option which may be used to replace the on-card, 8-channel multiplexer.

The 88-ADC will be available within 60 days of order placement at $524 (assembled only).

For further information contact: Mits, Inc., 2450 Alamo S.E., Albuquerque NM 87106.



Computer Kits new auto-load board.

NEW AUTO-LOAD BOARD

A new auto-load board, compatible with the Altair 8800, is now available under the trademark name of Power-Start, according to Pete Roberts, president of Computer Kits, Inc. and inventor of the device. Roberts said the Power-Start auto-load board can be purchased from Computer Kits either fully assembled or in kit form, with or without its own read-only memory (ROM). He said that Power-Start will soon be available from other computer stores throughout the country through a marketing agreement with RMQ Systems.

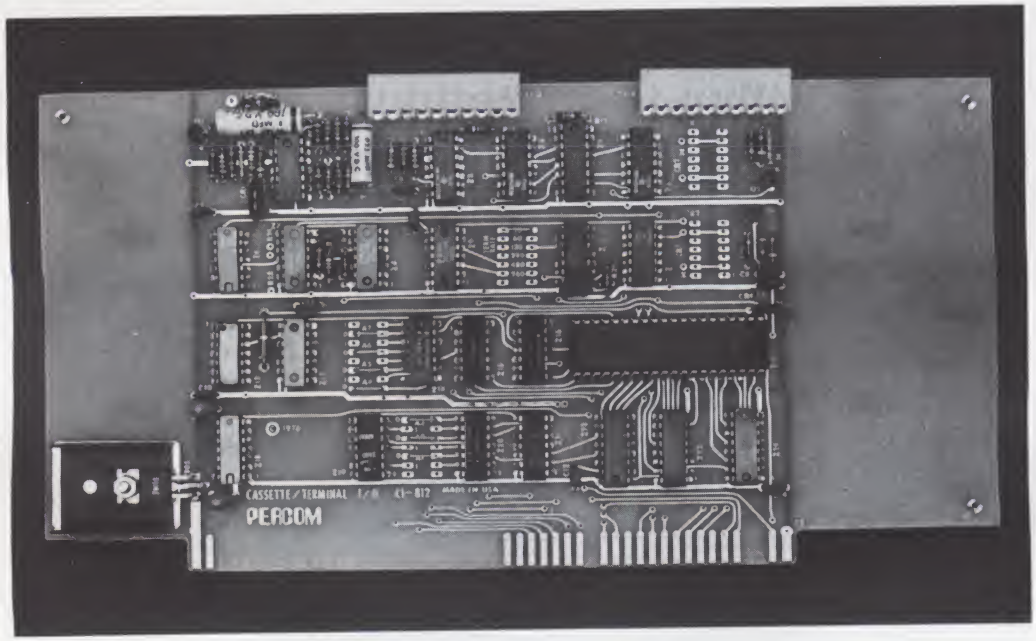
Basic advantages of Power-Start, Roberts said, are that it eliminates keying in of bootstrap programs; eliminates the need to reset sense switches; and allows the Altair 8800 to operate without a front panel. "To load the computer using the Power-Start," Roberts said, "the user merely presses 'reset' on the front panel. This will automatically load the computer from disk, cassette, paper tape, or any other device."

Because Power-Start has on-board switches that simulate the computer's front panel sense switches, terminal options need be set only once. When utilizing OEM or turn-key systems with the Altair 8800, all that is required to use Power-Start are on/off and reset switches rather than a full front board.

Roberts said Power-Start requires no computer rewiring. It plugs into the backplane and has an on-board switch which allows it to be cut out when desired. The auto-load board can be used with the purchaser's own ROM or programmable read-only memory (PROM). It can be configured to execute a loading program anywhere in the computer's memory address space. Further, Power-Start can be configured to utilize BASIC or any other programming system.

Prices for Power-Start range from $145 for the basic kit to $295 for the fully assembled auto-load board with ROM.

For further information contact Larry Glazier, Lewis and Associates, 68 Post Street, Suite 506, San Francisco CA 94104.



PerCom CI-812 dual function card.

CASSETTE AND RS-232 INTERFACE ON A SINGLE ALTAIR BUS CARD

PerCom Data Company has introduced the first Imsai/Altair compatible, dual cassette/terminal interface card. Designated the CI-812, the dual function card combines interfacing functions normally requiring two or three PC cards.

The cassette interface phase encodes (Manchester/Biphase) at the KC standard rate of 30 bytes/second, and at 60, 120, or 240 bytes/second for rapid loading of frequently used programs. In fact, the CI-812 is the only interface on the market today which provides both KC standard and high speed phase encoding.

The advantage of self-clocking encoding is that users can expect extremely high reliability, even at the fastest data rates, using simple, inexpensive audio cassette recorders. The self-clocking feature virtually eliminates tape speed variation errors.

The cassette interface record and playback circuits are completely independent, and the card is patterned to include optional DIP reed relays – which may be ordered as a kit – for program control of two recorder/players. This permits operations such as cross-filing.

The CI-812 companion circuit, the RS-232 terminal interface, is full duplex and provides for data exchange at 300 to 9600 baud. The CI-812 has been designed to operate with existing user's software with little or no modification. The CI-812 kit price is $89.95. Assembled, it costs $119.95. An instruction manual is included.

For further information, contact PerCom Data Company, 4021 Windsor, Garland TX 75042.

Letters to the Editor

COMPUTER-CONTROLLED SERVICE STATION

I am a satisfied customer (subscriber) to your fine magazine which contains many very interesting articles on microcomputers and solid state technology. I am also a ham WAØLKE who is very interested in learning how to build the many IC projects outlined in your magazine.

Speaking of IC projects, I have enclosed a couple of service station *self-service* transaction systems brochures. If I had the smarts, I would dearly love to build these devices myself for use in my service stations. I could save a bundle and also apply the kind of technology you guys express so well in your magazine.

I am sending you these brochures in order to ask your advice in how to go about building a similar system myself. Should I try to cotton on to some schematics or attempt to design from scratch? Hope you can be of some help and look forward to hearing from you soon.

Bob Bunn
West Plains MO

I've mentioned several times in the I/O Report that it would be a good idea if we all started looking for dedicated controller applications for microprocessors. The applications are almost unlimited and the money to be made isn't small! Unfortunately, Bob, I suspect the application you're talking about (service station control systems) would fall into the heavy category. Perhaps we should work our way up to something like that? If you're going to attempt a design from scratch then let me suggest you get a home system and get familiar with programming, and what it can do, before taking on a project such as this. – John.

GROUP PURCHASE CAN BE VERY EXPENSIVE

I found out the dangers of group purchase the other day when a letter came from the SCCS telling the following sad story:

Dear Group Purchase Member:

On October 6, 1976 the Board of Directors of SCCS suspended group purchase for a audit. It had been learned that a prepaid $9,000.00 order would not be delivered and the dealer might go out of business.

On November 10, the Board of Directors voted a refund to be given this month for 20% of any funds a member had in group purchase. Several methods of recovering the advanced funds are being pursued.

I was one of the lucky members who was returned 100% of my money due to the fact my check was received by the SCCS after the suspension of group purchase for the audit.

I got off easy, but I hope when you consider group purchase, you consider what almost happened to me and did happen to some of the SCCS members.

Randy Fallgatter
Goleta CA

MICROPROGRAMMABLE MACHINES

I would be interested in articles about bit-slice microprogrammable microprocessors – what types are available, how they are interconnected to form a system, how they are microprogrammed. I have some ideas for special-purpose CPUs that could probably be easily implemented with bit-slice microprocessors, but I find that I need some general background information before I can clearly understand the manufacturers' literature.

I wish you the best of luck with *Kilobaud*.

Alan R. Saminsky
Bethleham PA

DOS USER'S GROUP?

Please enter my subscription to your magazine. Enclosed is a photo of my system (Imsai 8080 with dual floppies). I would like to correspond with other DOS users. Thank you.

Eugene Christianson
Santa Barbara CA

You probably ought to watch out for this one (I know him). After all, can you really trust anyone who would wait until the first issue was out before subscribing to the magazine? (Just kidding. Gene is definitely one of the white hat good guys.*) – John.*

THE BIG GUYS & US

I've just finished a year teaching computer science and working with the school computer system, a very conventional IBM 370 with virtual storage, complete with all the systems programmers it takes to speak to it. On the basis of that, I've a rather curious opinion of the relationship between hobby microcomputing and the establishment: There have been more improvements in computing in the last two years, due to hobbyists, than IBM has made in the last ten!

An example: The virtual storage OS tells the FORTRAN compiler that his elbow room for symbol tables is big . . . 400K or so. As a result, the compiler hashes into that size array, but since pages are only 2K long you end up with about one table entry per page. Run time goes to pot. I had the systems man tell me with a straight face that this was because the compiler was so smart. I said I'd hate to meet the dumb one!

Another: The FORTRAN compiler has the capability for writing onto disk. All it takes, aside from the write statements, is a DEFINE FILE statement in the main program. Oh yes, you also need about eight JCL cards to tell the stupid system what the compiler already knew about file definition, and if the definitions don't match, you're dead. Oh yeah, one other thing . . . you won't be able to write into the file until you've first *erased* it, so you need to call a program to do that. That program also needs about eight JCL cards, but *not the same* ones, and again if the cards don't match . . .

The point is this: Hobby computing has progressed in two years to the point where we would never stand for such nonsense. We *want*, not *demand*, that we be able to enter the room, turn on the power, invoke a PROM compiler, and be running in minutes. To think IBM's customers have been willing to put up with that crap so long! On that basis, I'm going to make a prediction . . . MITS will put IBM out of business! Well, maybe not completely, and not right away, but it will happen. Remember when all radio-phonographs were made by RCA or Magnavox? Then along came those garage operations for audio freaks called AR, Dynakit, Garrard, etc. How foolish of them to think they could compete with the capital and engineering know-how of RCA! OK, I'll admit RCA is still around, but only because they went into TV. And now we have Advent . . .

Well, that's it for this trip. I know you're wondering when I'm going to write that article, and in the time it's taken me to write the letters I've sent I could have done it.

Dr. Jack Crenshaw
Huntsville AL

IT'S IN THE "T" REGISTER, PETE!

I have read and reread the contents of Issue #1 of *Kilobaud* and thoroughly enjoyed it. The selection and subject matter has been very informative which is just what a neophyte like me needs.

I am sure many readers are in the same boat with me wanting to see more informative and educational articles like Peter Stark (Programming? It's Simple, page 86). Having digested every word I did find what I believe is an error in Fig. 4 on page 89. Peter takes you down the right (or YES) side of the flowchart and calculates the overtime pay with the excess over 40 in the T register. After calculating the pay for overtime hours which is placed in register 2 he then recalls the number of hours from register 0. This is an error. Register 0 contains the total hours (i.e., 40 plus the overtime). The instruction should read "RECALL NUMBER OF HOURS FROM REGISTER T" (which is 40).

Of course, the above was a minor error because the value of the article was not lost. If I had been reading his lesson with less than enthusiasm I would have probably not noticed. So keep up the good work Peter and give us all some more.

As for *Kilobaud*, "keep it coming". I am eagerly looking forward to the next issue.

John T. Craig
Gig Harbor WA

Gig Harbor, Washington. Hmmmm. My Dad lives in Gig Harbor (same name as mine). Perhaps you know him? – John.

Letters to the Editor

COMPUTER STORES

During the past month, I have visited three computer stores, one in Atlanta and two in Tampa, and have found them to be totally different from the cold and formal business establishments I had thought them to be.

Conditioned by hundreds of visits to radio parts stores, I was surprised to find that computer shops had no counters attended by inattentive salesmen who periodically answer telephones and disappear behind rows of shelves and parts bins.

I found, instead that the computer store salesroom displayed operational computer systems that were available for *hands-on* use by visitors. Also on display were various makes of input/output devices. Salesmen in attendance were, generally, young, informal and knowledgable about the hardware and software capabilities of the equipment they handled.

Admittedly, my own sampling of computer stores is now small. However, by mid-February, I expect to have visited other computer stores in San Francisco, Honolulu, and Las Vegas.

Perhaps there are many other prospective computer hobbyists who would visit computer stores if they knew that they could view and evaluate competing microcomputers in a showroom atmosphere. Perhaps, too, those who presently hesitate to undertake a computer kit project might be encouraged to try after receiving assurances of success from technically competent salesmen at their nearby computer store.

If you feel, as I do, that readers of *Kilobaud* might be interested in an article that offers a vicarious visit to a hobby computer store, please give me a go-ahead to prepare such a piece. I expect to be able to submit with the article a number of 8 x 10 B&W glossy prints depicting the atmosphere of a typical computer store showroom.

Sherman P. Wantz
Sebring FL

That's a definite go on the article on computer stores. Sounds like you'll be bringing out some interesting observations and things people should look for. – John.

TYPESETTING PROGRAMS

First, I'd like to congratulate you on the first two issues of *Kilobaud*. Everything I've overheard at various computer hobbyist gatherings has been strongly positive. Didn't take you long to outclass most (all?) of the competition, did it?

There's one problem in the "Useful Loan Payment Program" that Phil Feldman and I wrote (Issue #2, page 68). Line number 200 of the program listing should read

200 C=(1+R)↑M

In the published listing, the up arrow was omitted.

When we proofread the copy, we saw that an up-arrow was already hand-drawn in by someone, so we made no additional mark ourselves. Should we have?

This brings up a point that you have already given careful consideration to, I'm sure. It seems that typos are going to be a definite problem in program listings as long as you typeset them. We'd be happy to provide good, clear, *camera-ready* copies of our program listings so that this problem could be avoided. When you realize that one or two small errors can totally change the meaning of a line of code (e.g., a colon instead of a semicolon, or an 8 instead of a B), you can see how serious this can be. A program that won't work isn't too useful.

A related problem is size and type of characters being used. The print is a little too small to read easily, and some of the characters are especially difficult (colon, semicolon, and the short minus sign in particular).

In short, you're doing almost everything right. With a little improvement in the quality of program listings, you can remove the word "almost" from the previous sentence.

Tom Rugg
Los Angeles CA

You're right, Tom, we did give it careful consideration and decided that we didn't want Kilobaud *to have the appearance of some of the other magazines who use* camera-ready *listings. It looks awful. On the other hand, I can certainly appreciate the value of using the computer's output. We're going to do everything possible to take care of the problem with improved proofreading. We're experiencing "growing pains" in one or two areas. This is one of them. – John.*

A ROBOT TYPIST?

I received your brochure this morning, and my first thought was not in favor of a new magazine, if you want to know the truth. But I read how you wanted to make yours more understandable for beginners, and if you can do better than the others, you will have a magazine that offers an advantage. The price, however, is steep for a yearly subscription, unless one has a good income. I have an idea that the beginners in this field are people who haven't had the money to get into it, and the wealthier ones who can afford such equipment are in it already, so that means that unless I am wrong, you may find it a bit difficult to sell subscriptions. They are more likely to buy on a single copy basis at the newsstands, but even so, the $2 price will keep them from buying every single issue unless they get interested.

Another thought occurred to me, which may be helpful, and that is the name of the magazine itself. No beginner is going to know what it is. *Kilobaud* is a complete mystery to anyone not in the field. You would do much better to have a name with Computer in it, which would tell at a glance what to expect inside. I think it is still not too late to change the name of the magazine, if I were you, and you ought to do it if you want to appeal to the public who don't know anything about computers. But the public is fascinated and might buy an issue to see if they can learn a little bit. What you have inside will either keep them reading, or it will turn them off.

In my own case, I investigated and found the subject interesting, but the equipment was too expensive, in spite of the come-on advertising that one could buy an Altair for $399 when it was first offered. The trouble was that I expected the price to get lower, but it went higher, and I learned by that time that the basic price was only the beginning. Almost each addition ran $115 to $175, and there was no end of such costs, just for the cards. Then there were the peripherals, and these were even higher.

I didn't even know what I would need, luckily, and had to learn enough just to know that I couldn't afford to get involved. So I took my money I had saved by the time a year went by, $550, and bought this typewriter, and if anyone can ever figure out an inexpensive way to hook up an IBM Executive with Testimonial type and proportional spacing to a Sylvania Color TV via an Altair or Imsai with robot typer to retype my written material, I might be interested.

Don F. Hill
Hemet CA

P.S. If this letter is any help to you, throw my name in your sweepstakes. That's the only way I would get started.

GETTING UP TO SPEED

You asked for ideas for articles for *Kilobaud*, so here's mine. I'm thinking of an article or series about Joe Kilobaud who's thinking about getting a hobby computer and develop it as a useful one after learning all about it; however he can only part with, say, $50 – $100 at a time without major difficulty. Obviously he wants to get started on the first purchase, but he wants to eventually have a full size computer with all capabilities (whatever those are) that is easy to expand and update and will be versatile. So let's say his first buy is a microprocessor with power supply, lights and switches. After learning about machine language, he decides to go with BASIC since it's most popular and much faster than machine language. On his next purchases, he adds such things as TV terminal, typewriter keyboard, memory or expansion, ability to phone connect to a large computer out of town, etc. The article explains Joe's thinking as he considers what objectives people have in mind when they choose various CPUs such as the 6800 or 8080A which have gained widespread use and are likely to have *spare parts* around for a while. (Here's where I as a reader learn about speed vs. efficiency, of instructions, 8-bit vs. 4-bit vs. 16-bit units, memory considerations, I/O ramifications – serial vs parallel, and cost vs. performance tradeoffs. I learn what basic units I must have to be functional at all, and the logical order of expansion depending on what I want to accomplish.)

If I knew the answers to these proposals, I'd write the article myself; however, I've "read about enough to be dangerous" and have stated what I'd like to know so I can get started on my own hobby computer and expand it to meet the needs of my own business as well as personal needs. Essentially, the article would be a basic course on what the microprocessors can do, what I need to get started at a minimum startup cost, and what I need to know to make intelligent decisions and selections. (*The Popular Electronics* December issue on this topic generated this letter.)

Also, a friend at work (instrumentation engineering) at Buick Motor Div. subscribes to 73 *Magazine*, and the I/O section on using ICs were of particular interest and help. I'm thinking of subscribing to it as well if it will keep up those articles – I don't have time to develop an interest in ham radio, so if it now becomes how to use ICs in ham gear, I'd have little interest. However, if general applications of the ICs continue, I'll subscribe as well as get some back issues (*back to June 75*) for me and my brother. If you've got time, I'd appreciate a word from you about this. I'm wondering if the January issue is answering this question, with all the ham articles.

In all, the *Kilobaud* magazine looks very promising. I hope it's able to fulfill that promise. I know you and the staff will be trying.

John Nierste
Clio MI

Thanks for the good ideas, John. And, as far as the I/O section of 73 Magazine *is concerned, it does look like the majority of the articles will be dealing with ham radio applications for microcomputer systems. – John.*

continued on page 73

Interrupts Exposed

. . . using microprocessor interrupt capability effectively

Dan La Dage
PO Box 28517
San Diego CA 92128

Computer club meetings can be a lot of fun, especially if you have a chance to travel and can visit different clubs. I enjoy the gossip, the rumors of new products, and the esoteric trivia valued only by the dedicated hobbyist. There is always a surprise for me at these meetings, always something new to learn. On one visit recently someone had set up his home brew system in the back of the room and it was attracting a lot of attention. The builder reached into its innards and jiggled something and suddenly the screen went blank.

"What happened?" someone asked.

"Oh, I just interrupted it back to the monitor," was the reply.

"What's an interrupt?" the observer asked, wanting to find out everything about that beautiful tangle of lights and wires.

"That's when the computer stops executing your program where it is and starts executing somewhere else."

"It does *what*?" the questioner replied, and then was silent as he waited expectantly.

Interrupts happen to the best of us, and they can happen at very inconvenient times. I sometimes think they *only* happen at inconvenient times, just like the telephone that only rings when I am in the shower. A knowledge of how they work can quell our uneasiness, however. And in systems which have real-time capability, it is essential that we know precisely how they work and how to utilize them to make the system run properly.

For example, consider a home fire and security system which has all kinds of remote sensors. The sensors may be smoke detectors, window and under-carpet switches, or sound or proximity detectors. When a sensor is activated, it sends a signal to your computer, which can take any action called for — call the fire department, ring bells or turn lights on, call the sheriff or take a picture of the

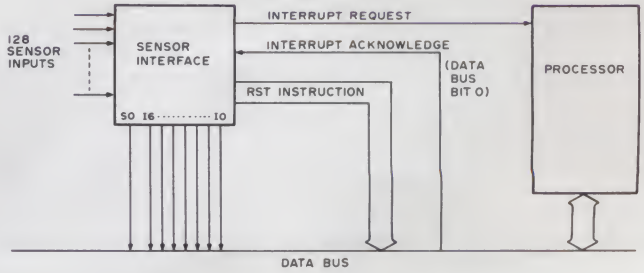


Fig. 1. Hardware interface for sensors.

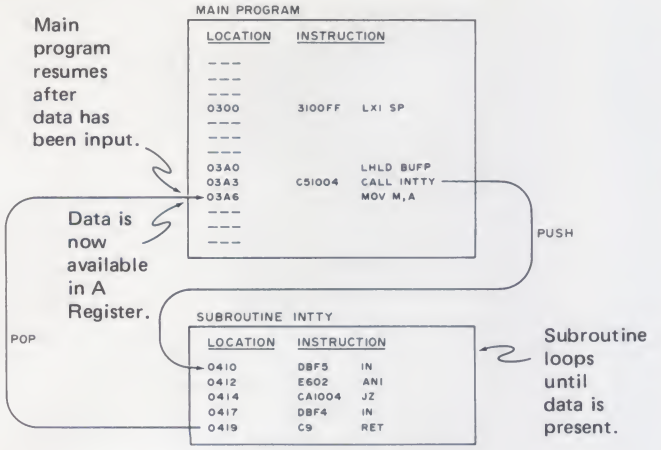


Fig. 2. Subroutine call/return structure.

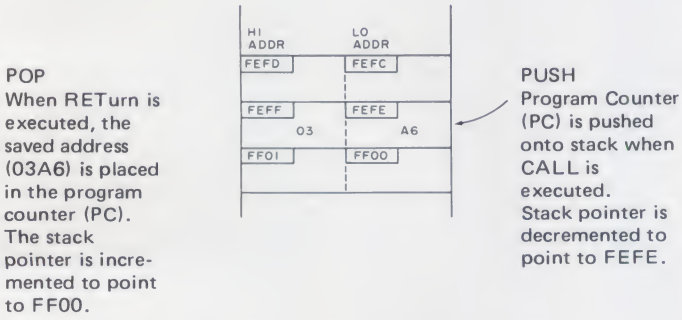


Fig. 3. Stack contents during program flow.

intruder. Fig. 1 is a block diagram of such a system. You can dream up all kinds of response actions. Our purpose here is to make certain your program handles the interrupt properly. But before we can do this it is first necessary to understand what happens in normal I/O, what happens in interrupt-driven I/O, and how and why they are different.

Three Ways to Input Data

From the software point of view, I can do *in-line I/O, polled I/O,* or *interrupt I/O.* For in-line I/O, when I execute an input instruction, the computer simply stalls until the data has been input, before continuing with the next instruction. This is not common in microcomputer systems, but it is standard practice in higher languages such as BASIC. In this language we might write:

```
130 X=A+3
140 INPUT Y
150 X=X+Y
```

Statement number 150 will not be executed until Y has been input. Of course we know that the computer is still running — the BASIC interpreter or compiled code is cycling, waiting for the data to be input. But from the BASIC programmer's point of view, statements 130, 140, 150 are executed in sequence, and the computer appears to wait at statement 140 until the data has been input. Old-fashioned tube computers used to run this way, even in assembly language, when it was too expensive to build them differently.

In hobby computers, the common way of performing input is by polling. You can see this in the sequence of 8080 code in Program A. The program continually reads in the status until it recognizes that data is present in the data-port-buffer.

Of course, we don't need to loop indefinitely like that. If no data is present, we could go away and do something useful and then test the status later. But we have to remember to come back and try again. When we have to poll the I/O ports to test for data present, we have a responsibility to keep at it until the data comes in. So sometimes it is simpler in the long run to just loop and wait. Notice that if INTTY is used as a subroutine, then we have actually simulated in-line I/O (see Program B). The CALL instruction acts like an in-line I/O statement. That is, the MOV instruction at 03A6 will not be executed until the data has been input.

When a subroutine CALL instruction is executed by the processor, this is what happens:

1. The location of the next instruction after the CALL is placed in the memory location addressed by the stack pointer (SP).
2. The stack pointer is decremented by two.
3. The memory address field in the CALL instruction is placed in the program counter.
4. Execution resumes at the new location.

The stack is an area in memory that is used to hold these saved addresses. If we do any stack operations (like CALL and RETurn), we must have some area set aside to hold the stacked information. Also, we must initialize the stack pointer to point to this area, and we can do this with the SPHL instruction (move registers H and L to the stack pointer). Because the stack goes *backward* in the 8080, we should initialize the pointer to be at the highest address of the stack area. The stack can be initialized in the way shown in Program C. Fig. 2 illustrates the flow of control in a subroutine call, and Fig. 3 shows the contents and pointer values at each step. After the subroutine INTTY is CALLed, the stack contains the return address 03A6 and the stack pointer has been changed from FF00 to FEFE. When the RETurn instruction is executed, the contents of the stack (03A6) is put back into the program counter and the stack pointer is incremented to FF00. Execution then resumes at location 03A6. At this point, the data has been input from the device and is available in the A register.

Interrupt I/O works quite differently. With interrupts, the computer gives me the data before I even ask for it! No matter what my program is doing, the computer stops executing at that point and starts executing somewhere else. This is a rather brutal way to get my attention, but that is what interrupts are for. Suppose that I have been interrupted, and now suddenly I am executing at a new place (at the *interrupt-service-routine*, usually called the *ISR)*. The first thing I need to do is to find out who caused the interrupt. If I only

```
INTTY:  IN    TTST    ;Input Teletype status
        ANI   INFLG   ;Check for data present
        JZ    INTTY   ;Loop if not ready
        IN    TTDT    ;Input Teletype data
        RET
```

Program A.

```
03A0   LHLD   BUFP    ;Load buffer pointer
03A3   CALL   INTTY   ;Call the input subroutine
03A6   MOV    M,A     ;Store character away
```

Program B.

```
        LXI   H,FF00H ;Load address into HL
        SPHL          ;Move HL to stack pointer
```

Program C.

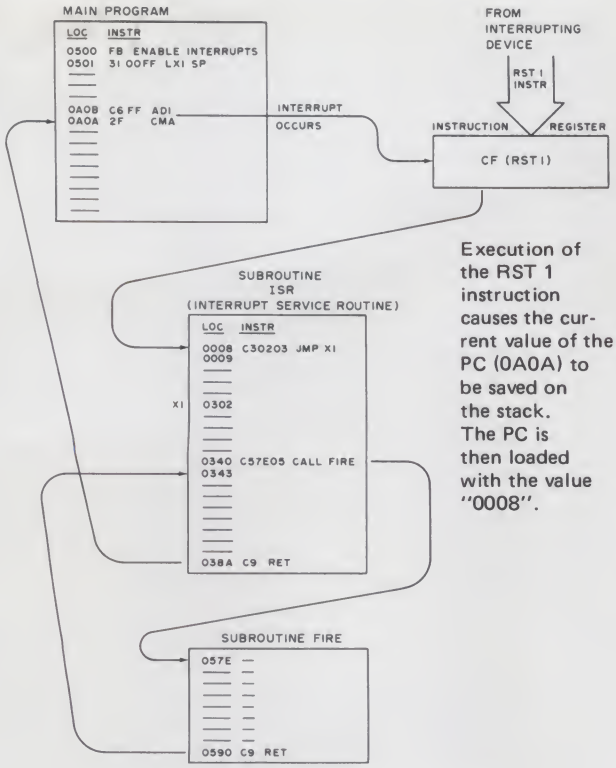


Fig. 4. Program flow in interrupt routines.

have one device that can do this, then it is trivial. But, since my home security system has lots of sensors, I would like to recognize them individually. This is a requirement in interrupt-based systems. The device, or its interface, must transmit to the computer some identifier. The computer can use this identifier in different ways — it can sort it in memory, or in a register, or leave it in a known port buffer, or it can use the identifier to index (form an address) into a table of jumps to different routines *(vectored interrupts)*. No matter how it does this, the computer must get this information from the device and pass it on to the program in some form.

Security System Hardware (And Software)

Going back to Fig. 1, let's take a look at the interface board for our security sensors. We want it to monitor a number of bi-state lines and to signal the processor when any one line changes state. The nice thing about our sensors is that they are binary; they are either on or off, open or closed. We could use one bit to encode the device state and seven bits to encode its identity, assigning a unique number to each sensor. These seven bits would permit us to have up to 128 devices per interface board. Of course we might not have room for all those wires, so we might have some physical limitations. But for now, let's just consider the logical structure. The lines from the sensors come into the board from outside. The board is connected to the processor by an interrupt line, and to the data bus by eight data lines. (There will be additional lines for control and clocking.)

The interface board will monitor the sensors, and if any one changes state, it will place its new state condition in S0, encode its identifier in I0 through I6, and place this information into the port buffer on the external bus. For an 8080-based system, the processor requires that the control signal be accompanied by an instruction for it to execute. We could choose any instruction, so let's pick the Restart (RST) because that will interrupt the software. How all of this works will be described later, but the reason for this "instruction" is that the computer and the interface board engage in a little dialogue about what should be done at an interrupt. The interface board signals the processor that it has an interrupt. The processor responds, in effect, "Okay, now you have my attention. I will grant you one wish. I will execute one instruction on your behalf. It can be any instruction, but only one instruction." And our interface board responds, "Here, take this. It is an (RST 1) instruction."

Now what happens on the inboard side? An "interrupt" from the board forces the processor to execute the (RST 1) instruction, which interrupts the current program and jumps to location 8. Then the software must read in the data from the port-buffer, and take appropriate action. It can read in the data with an IN instruction, getting eight bits into the A-register. The low-order bits (bits 6 to 0) will identify the device (sensor) and the high-order bit (bit 7) will hold its state (zero or one).

Let's take that sequence in a little more detail, so we can see how the interrupt works from the software point of view. First, the processor forces the program counter (the PC) to change from its current value to the value of 8. This is great for the interrupt, but eventually you will want to return to your program, so it is important that the processor should save the old value of the program counter so you will know where to return. All computers do this differently, but in most microprocessors the PC is saved in the stack. This is just what happens in the RST instruction in the 8080, which is a "CALL" to a specific location.

The RST instruction in the 8080 is a special CALL instruction, where the address of the destination subroutine is implied within the instruction.

```
RST 0 is the same as CALL 0H
RST 1 is the same as CALL 8H
RST 2 is the same as CALL 10H
...
RST 7 is the same as CALL 38H
```

The RST instruction does one thing which the CALL instruction does not do: it disables interrupts. This is, of course, all well and good because we don't want another interrupt to be recognized while we're in the process of servicing one.

Interrupt Request and Permit

In the 8080 microprocessor, there is an *interrupt request bit*, and an *interrupt permit bit* (called INTE). The interrupt request bit reflects the condition of the interrupt request line, which is connected to the processor from an external source. The interrupt permit bit is internal to the processor, and is set and reset by software or by the processor itself. After the microprocessor executes any instruction, it interrogates these two states. If INTE is =1, and the interrupt request line is active, then the processor accepts an instruction from the interrupting device and executes that one instruction. It also turns INTE off, resetting it =0. The processor bit INTE (the interrupt-permit bit) can also be set and reset by software. The DI (Disable) instruction turns the bit off (sets it =0), and the EI (Enable) instruction turns it on. If interrupts are disabled (INTE=0), then the processor will not even respond to the external interrupt request. In some 8080-based hobby systems, the RESET switch on the console is wired in as an "external" device with an interrupt line; it presents the processor with an (RST 0) instruction. In this way an interrupt acts like a forced CALL instruction to a

specific subroutine. From the point of view of program execution, it looks as though this (RST 0) instruction had been stuck into the middle of the program at some arbitrary point.

Figs. 4 and 5 show how this works in the running program. When the interrupt occurs, it forces the execution of the RST instruction, which pushes the location of the next instruction to be executed (in the example, 0A0A) onto the stack and transfers control to location 8. Another CALL within ISR will cause another location (0343) to be pushed onto the stack, and control goes to 057E. When the RETurn is executed, the contents of the stack is put back into the program counter and the stack pointer is incremented to FEFE. When subroutine ISR returns, the contents of the stack (now 0A0A) is placed into the program counter and control returns to the location following the place where the interrupt occurred.

The Security System Interrupt Operation

We have already discussed how the sensors might be connected to the interface board. Now we will consider how this board interfaces to the computer itself. Let's fix it so that when it interrupts the computer it presents it with an (RST 1) instruction to interrupt the software. It also must be able to tell the computer which sensor caused the interrupt, so we will have it put the device identifier into one of the I/O port buffers. Most conventional devices use two such ports: one for a status byte, and one for a data byte. We will use only one port — to hold the device state and identifier. The action of the interface board now looks like this:

1. The board monitors the sensors.
2. If a sensor changes state, the interface takes the new state value (0 or 1) and the identifying number of the sensor and concatenates them into an 8-bit number.
3. It places this 8-bit number in the I/O port buffer connected to the external bus.
4. It places an (RST 1) instruction in the interrupt-buffer for the microprocessor.
5. It activates the interrupt request line to the processor.

The processor:

1. recognizes the interrupt request (if interrupts are enabled);
2. reads in and executes the (RST 1) instruction. The rest is up to software, which will:

a. Explicitly read in the data from the I/O port, which contains the sensor state and ID.
b. Extract the state and ID and take "appropriate action."
c. Return to the main program that was interrupted.

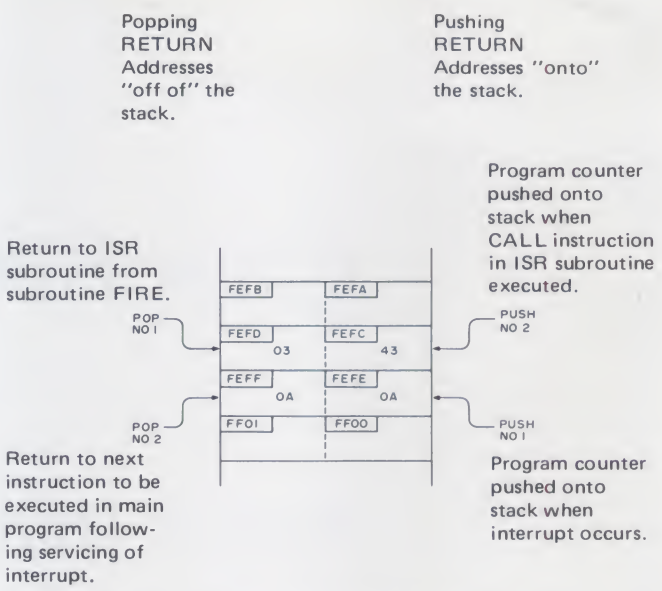


Fig. 5. Stack contents during interrupt processing.

What is this "appropriate action"? Our sensors can be any kind of binary on-off switches. We have thought of them as security sensors — like window leaf-switches or smoke sensors. But they could really be any kind of binary devices — thermostat controls, humidity or water-level sensors, or light or sound-level actuated switches. There are many uses for our interface board, and we could have many different devices attached to it. Most applications will also require control signals to be *output* from the computer, but for now we are only looking into the input aspect — the data-collection. We can simplify the control aspect by assuming that when we sense an unexpected event we can ring an alarm or cause an automatic dialer to place a call to the police or the fire department. If our system has mixed fire and security sensors, then we must be able to tell which is which, and we can do this easily in software.

Conclusion

Next month we will develop the software — the programs and the data tables — that will be required to process the interrupts and the data from the sensors. There are also some special difficulties in writing software for interrupt-driven systems. I'll discuss the most common problems encountered in real-time software and help *you develop* practical ways to deal with them. ■

from page 8

role. The addition of the requirement to act (make decisions, push buttons, more joysticks) reduces the amount of processing the user can do on the input and so has the effect of "heating up" the medium. Thus, if I've interpreted McLuhan right, his principles would predict that that acceptance of computers in the home would move the society back toward pre-TV values. It would lead us to expect a swing of the pendulum back toward the "rational" from the "mystical," toward specialization from global emotional involvement, toward a concern for content from a concern with superficial form, toward urban values from village-like values, etc. Perhaps we could take this so far as to predict an upswing in the popularity of other hotter media, like books and (ahem) magazines, and a decline in the popularity of passive TV (network TV as we know it now).

You can take these ideas further by reading McLuhan's characterizations of hot vs. cool societies (in *Understanding Media*), but let's not forget that there are quite a few *ifs* in front of these predictions. It is conceivable that home computers in any form won't achieve mass popularity, and in that case we'd expect their impact to be relatively small. It is conceivable that the effects McLuhan associated with different forms of media are really due to other factors (economics, dictation by powerful corporations come to mind). Finally, the trick of ignoring the content may fail us — perhaps home computers will be accepted, but only very cool, myth-like games like simple versions of Star Trek will be widely popular. At least it is fun to think about.

Several people have expressed an interest in devoting a future Lookahead to a collection of people's wildest fantasies about personal uses of computers. What will individuals be doing with computers in 50 years? If you have any thoughts on the matter, send them in and I'll collect them until we have enough for a column.

Write:

Lookahead
1218 Broadway
Santa Cruz CA 95062

[1] H. Marshall McLuhan, *Understanding Media: The Extensions of Man*, Signet Books, 1964. H. Marshall McLuhan and Quentin Fiore, *The Medium is the Massage*, Random House, 1967.

[2] Gerald Emanuel Stearn (ed.), *McLuhan: Hot and Cool*, Signet Books, 1967.

Don Lancaster
SYNERGETICS

Clocked Logic

... Part 2: Some basic applications

Last month Don Lancaster presented us with a good introduction to the world of flip-flops. His discussion this month covers devices such as basic counters, dividers, shift and storage registers, and multivibrators. In this second of a three part series he has taken material from his upcoming book entitled CMOS Cookbook *(to be published by Howard W. Sams). — John.*

Let's discuss some of the many different things we can do with the basic clocked logic D and JK flip-flops. These techniques are useful with the 4013 and 4027 by themselves or in simple circuits.

Most often, you'll probably only want to use a few 4013s or 4027s in your circuit as the fancier MSI blocks cram much more performance in a single package. If you find yourself using lots and lots of JK or D flops, try to find a MSI substitute or a different approach that will simplify the job for you. On the other hand, it's a very rare CMOS circuit that doesn't have two or three 4013s and maybe a 4027 tucked away in a corner somewhere to pick up some loose ends that the MSI can't handle directly. So it pays to be aware of all the different good things you can do with these basic clocked logic blocks.

Binary Counters

Binary counters are probably the oldest of clocked flip-flop uses. We can get a single stage to divide by two either by cross coupling $\overline{Q}$ to D on a 4013 or by making both J and K high on a 4027. The output alternates states, giving us a square wave with a 50-50 duty cycle of one-half the input clock frequency.

We can *cascade* binary counters as shown in Fig. 7. This lets us count to numbers higher than two or divide an input clock by a higher ratio. If the output of one divide-by-two (Fig. 7(a)) is connected to a second so its output clocks the second stage, we end up with the divide-by-four of Fig. 7(b). Add another stage, and we pick up the divide-by-eight of Fig. 7(c). More stages mean more possible count states and a higher division ratio. Four stages is particularly interesting. By itself, it can represent sixteen different things, count to sixteen, or scale an input frequency by sixteen. But, if we properly tamper with the count sequence, we can shorten our divide-by-sixteen into a divide-by-ten or decimal counter and do "by tens" counting and arithmetic.

These binary counters are called *ripple* counters. One stage has to change completely before the next stage can start its changing. Note

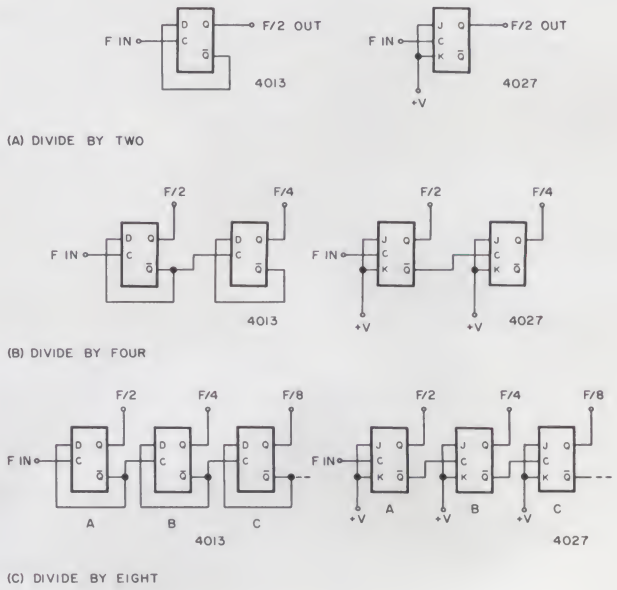
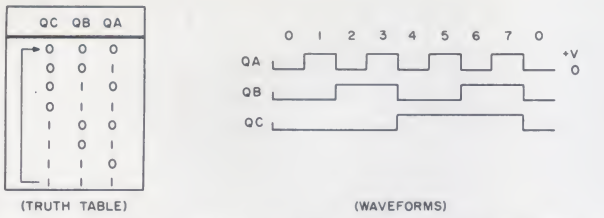
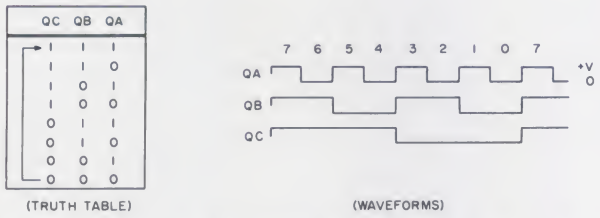


Fig. 7. Binary ripple counters.



(A) NORMAL, "ADD" OR UP-COUNTER. STAGES CHANGE ON HIGH TO LOW TRANSITION OF PREVIOUS STAGE.



(B) REVERSE, "SUBTRACT" OR DOWN-COUNTER. STAGES CHANGE ON LOW TO HIGH TRANSITION OF PREVIOUS STAGE.

Fig. 8. Binary counter waveforms.

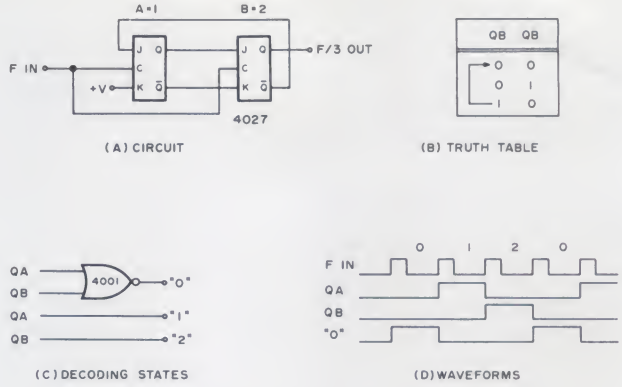


Fig. 9. Synchronous divide-by-three is weighted 1-2.

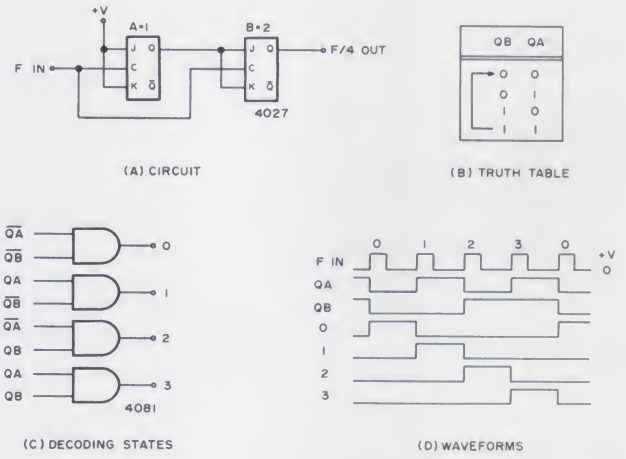


Fig. 10. Synchronous divide-by-four is weighted 1-2.

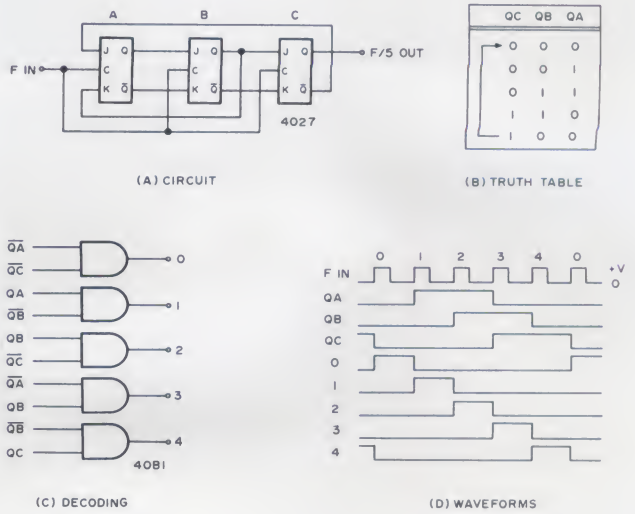


Fig. 11. Synchronous divide-by-five has 3:2 output duty cycle.

that invalid output counts will happen during the *settling times* caused by the stage-to-stage *propagation delays*.

We can control the count direction, depending on how we drive the clock of each stage. Fig. 8 gives details. If we clock from $\overline{Q}$ of the previous stage, we get a normal, *add*, or binary up sequence as shown in Fig. 8(a). On the other hand, if we use the Q output to drive a positive edge clocked next stage, we end up with a backwards, *subtract* or down counter, as shown in Fig. 8(b). If our logic blocks are negative edge clocked (such as the 4024), the exact opposite is true — cascade from Q for a normal or up sequence and from $\overline{Q}$ for a reverse or down sequence.

Divide-by-Three

Fig. 9 shows us a *synchronous* divide-by-three counter using a 4027. Note that both stages are clocked at the same time from the input, so we don't have the propagation and ripple delay effects of cascaded stages. The output of this counter is said to be *weighted* 1-2, meaning that one output counts for "1" if it's there and the other one counts for "2" if it is present. So, you can directly look at the states and immediately tell what count is stored in the circuit. This circuit is the shortest example of the odd length walking ring counter. Two of the three counter states are self-decoding; the third is picked up with the NOR gate shown in Fig. 9(c).

Divide-by-Four

A synchronous alternate to the ripple divide-by-four is shown in Fig. 10. We use the J and K low "do-nothing" state of a 4027 to inhibit the counting of the second stage half the time. Weighting is also 1-2. Four two-input AND gates may be used to decode the individual stages as shown.

This "do-nothing" inhibiting of a JK flip-flop is the key to longer synchronous counter. For a divide-by-eight, you only let the third stage count one-fourth of the time and inhibit it three-fourths of the time. A divide-by-sixteen can count only one-eighth of the time and so on. You can either use multiple input gates or a cascaded sequence of enabling two-input gates for longer synchronous counters.

Divide-by-Five

Here is another example of our odd-length walking ring counter. As Fig. 11 shows us, the circuit is synchronous with all stages clocked di-

rectly from the input. We can decode this particular circuit with five two-input AND gates as shown. The output is unweighted and has a 3:2 duty cycle.

Any of these counters can be reset to zero by using the Direct Reset inputs. You do have to be sure the direct input goes back low before the next clock pulse arrives. With combinations of direct set and direct reset, you can load any desired count into your circuit any time you want.

Shift Registers

A *shift register* is built as shown in Fig. 12. We cascade the Q output of a D flip-flop to the D input of the next stage. With JK flip-flops, we connect Q to J and $\overline{Q}$ to K, making sure the first stage always sees complementary data on the J and K inputs.

Each stage stores one *bit* of data, forming a *word* equal in length to the number of stages in the register. On clocking, each bit moves one stage to the right. The first stage picks up a new one or zero from the serial input. The last stage sends its output on to the outside world or loses it. The registers shown in Fig. 12 are usable as serial-in-serial-out (SISO) or serial-in-parallel-out (SIPO) registers. We can also build shift registers with parallel loading direct inputs, and, if we like, we can recirculate shift register data from output to input.

Storage Register

We can also use a pile of D flops all at once rather than having them pass data to each other. This gives us a *storage register* that accepts and holds a *parallel* word for us. An 8 bit parallel storage register is shown in Fig. 13.

Storage registers are useful to catch data on the way by, particularly from a microprocessor. They then hold the data as long as we need it. You can also use storage registers to sample data when it is known to be good, eliminating any intermediate garbage caused by settling times, propagation delays, and so on. A storage register on the output of an electronic music digital keyboard will hold the note command for us after key release. This lets the note decay and fall-back continue after the note is let go, still telling the rest of the circuit what note it was working on.

Some MSI examples of storage registers include the 4175 quad, 4174 hex, and 4034 eight-bit devices.

Monostable Multivibrators

A normal monostable multivibrator using the 4013 D flop is shown in Fig. 14(a). Clocking drives Q high, which charges C through the series combination of R2 and the much smaller R1. When the cycle ends, C is rapidly discharged through R1 only. Leaving R1 off gives very fast recovery but distorts the Q output waveform. If a long recovery time is available, we can use R2 only and omit the diode.

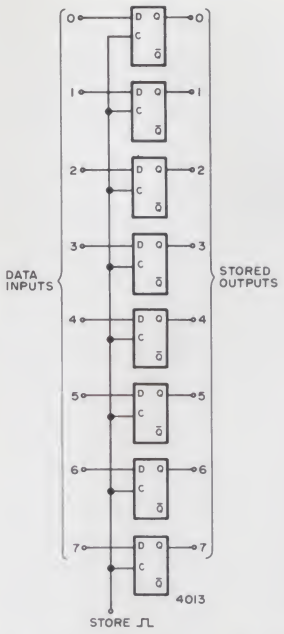


Fig. 13. 8-bit word storage latch for a microprocessor.

To pick up a retrigger ability, examine Fig. 14(b). Here the input clock low time discharges the capacitor through R1. The positive clock edge drives Q high and R2 charges C for the delay-until-reset time. The circuit may be triggered at any time and will time out from the last triggering. Note that the monostable ON cycle cannot *end* while the clock is low.

We can also use the alternate trigger method of Fig. 14(c). Here we pulse the SET input to start timing. This takes a resistor and a capacitor, but gives us a second way to positive edge trigger. The time constant on the trigger must be shorter than the ON time for proper operation.

The system reset/power-on generator of Fig. 14(d) will give you a clean reset signal shortly after power is applied to your system. Applying supply power triggers the monostable which then times out long enough for the supply to reach a stable value. The trailing edge of the monostable can then be used for a system reset. This type of circuit is handy for initializing things like microprocessors, making sure every thing comes up in a benign state when first activated. ■

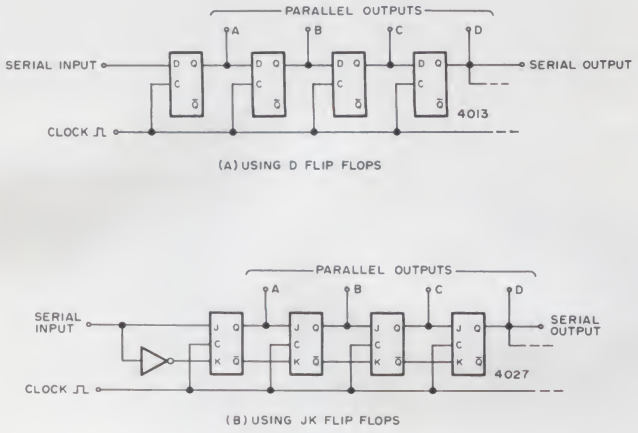


Fig. 12. Shift registers.

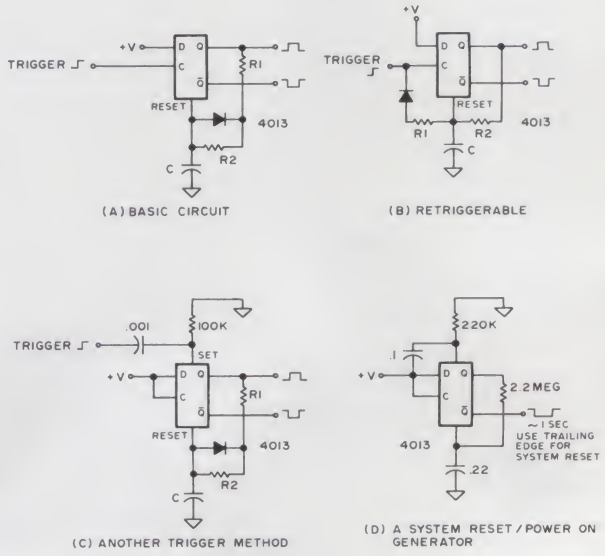


Fig. 14. Monostable circuits.

GET THE BEST FROM... SEALS

Let me get the best . . . Please send the following

- ☐ 8 KSC 500ns ☐ KIT $295.00 ☐ ASSMB'LD $349.00
- ☐ 8 KSC-Z 250ns ☐ KIT $295.00 ☐ ASSMB'LD $349.00
- ☐ WWC ☐ KIT $37.50 ☐ ASSMB'LD $47.50
- ☐ BBUC ☐ KIT $55.00 ☐ ASSMB'LD $68.00
- ☐ EXT Extender card $9.00
- ☐ 100 pin edge conn ☐ ALTAIR® $9.00 ☐ IMSAI® $9.00
- ☐ ASSEMBLY & OPERATING MANUAL $4.00

NAME _____
PLEASE PRINT OR TYPE

ADDRESS _____

CITY _____ STATE _____ ZIP _____

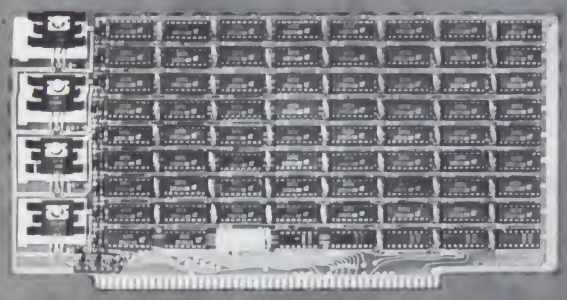
SEND CHECK • MONEY ORDER • COD'S ACCEPTED • CREDIT CARDS

BANKAMERICARD — COPY ACCOUNT NUMBER FROM YOUR BANK AMERICARD — MY CARD EXPIRES

master charge — COPY ACCOUNT NUMBER FROM YOUR MASTER CHARGE — Copy number above your name on Mastercharge — MY CARD EXPIRES

SEALS ELECTRONICS, INC
P.O. BOX 11651
KNOXVILLE, TN 37919

MOST ORDERS SHIPPED WITHIN 10 WORKING DAYS



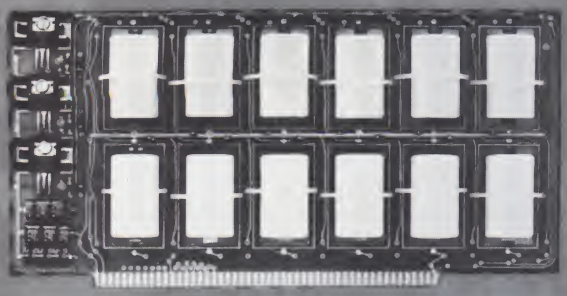
8 KSC [8K STATIC MEMORY BOARD]

Our most popular item. Hundreds of satisfied customers. We have received an enormous number of letters praising our 8 KSC board. Our 8 KSC is undoubtedly the highest quality and most dependable board on the market today.

INTRODUCTORY OFFER

We are proud to announce that you can order your 8 KSC-Z with 250ns memory at the same-yes SAME-price as our 500ns memory.

		KIT	ASSEMBLED
8 KSC	500ns	$249.00	$349.00
8 KSC-Z	250ns	$249.00	$349.00
EXT extender card		$ 29.00	$ 38.00
100 pin edge conn. [Altair®]			$9.00
100 pin edge conn. [IMSAI®]			$9.00
Assembly & Operating Manual			$4.00

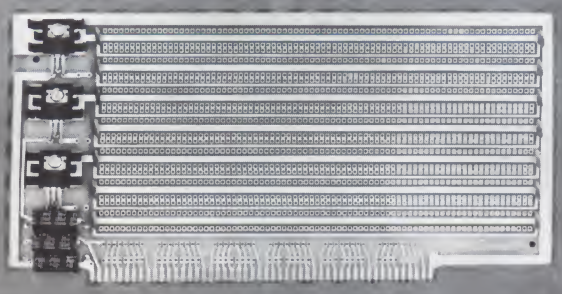


BATTERIES NOT INCLUDED

BBUC [BATTERY BACK UP BOARD]

- Automatic battery charging circuit
- Selectable standby voltage outputs
- Will hold up to 12 "C" cell Ni-cad batteries. As much as 12 Amper hrs
- The BBUC comes selected for 2.5 volts standby to pin #14 on the S-100 buss structure, to power up the 8 KSC memory
- Can be wired to back up any memory card which has battery standby capability. Even **TWO** polarities at one time
- Eliminate cluge wires on top of memory
- **Heavy G-10 glass epoxy PC board**
- **Heavy plated through holes-.5 mil. tin minimum**
- **Solder mask both sides**
- **Component layout screened on component side of PC board**

KIT - $55.00 ASSEMBLED - $68.00 ASSEMBLY & OPERATING MANUAL - $ 4.00



WWC [WIRE WRAP CARD]

- Accepts **ALL** IC wire wrap sockets 40, 22, 16, 14, etc.
- 3 voltage regulators: +12v, -12v, +5v
- 3 separate input capacitors 100 ufd
- 14 .I ufd decoupling capacitors

- Gold plated edge contacts
- **Heavy G-10 glass epoxy PC board**
- **Heavy plated through holes -.5 mil. tin minimum**
- **Component layout screened on component side of PC board**

KIT	ASSEMBLED
$37.50	$47.50

SEALS® ELECTRONICS, INC.

TELEX # 55-7444 twx # 810/583-0075 TELEPHONE # 615/693-8655 S-22

PRIME RADIX

PRESENTS 64K™

WE DO IT WITH MIRRORS!

(and some very sophisticated state-of-the-art memory design)

You've probably imagined that someday you'd like to own a computer system with a full complement of memory:

65,536 BYTES

Your dream can be a reality with the Prime Radix Corporation's 64K™ memory system at a very cost-effective price. And because it is a standalone memory system, you've got the advantage of greater flexibility not ordinarily available from add-in memory. Some of the features are:

- The 64K™ is fully buffered, presenting one TTL load to the memory bus.
- The 64K™ is digital group bus and ALTAIR™ bus compatible. **When ordering, you must specify the bus architecture.** A plugcard and cable will be furnished for the particular bus architecture you specify.
- The minimum complement of memory is 40K BYTES, with starting address locations at 0K, 8K, 16K, or 24K. Switchable memory protect is in increments of 8K bytes on 8K boundaries.
- The 64K™ comes assembled and tested with its own power supply, attractively housed in an aluminum cabinet, ready to plug into your system with a choice of a freestanding or a 19" rack mountable cabinet, 5" H x 18" W x 14" Deep.
- Psuedo-static operation: on board refresh clock-generator provides processor independent refresh with no wait states. The 300NS worst case access time enhances high speed operation.
- Power/fail detection circuitry and battery backup will provide non-volatile memory (batteries are optional at extra cost).
- The 64K™ has an expandable organization to other bit word lengths.

LIST PRICE IS AS FOLLOWS:

40K	48K	56K	64K
$1490.00	$1580.00	$1670.00	$1750.00

We are offering a special introductory ten percent discount off list price on all orders received on or before February 28, 1977. Delivery will be made in the same sequence as orders are received. Please allow 3 to 6 weeks for delivery. Mastercharge and BankAmericard are accepted.

• PRIME RADIX, INC. • P.O. BOX 11245 • DENVER, COLORADO 80211 • (303) 573-5942 OR 433-5630

PRIME RADIX
COMPUTER SYNTHESIS

☐ DIGITAL GROUP BUS
☐ ALTAIR™ BUS
☐ 64K @ $1750.00
☐ 56K @ $1670.00
☐ 48K @ $1580.00
☐ 40K @ $1490.00

☐ Check or M.O. enclosed
☐ Charge BAC
☐ Charge MC
(Please No C.O.D.'s or P.O.'s)

Make checks or money orders payable to:
PRIME RADIX, INC
P.O. Box 11245
Denver, Colorado 80211
(303) 573-5942 or 433-5630

Print Name

Address

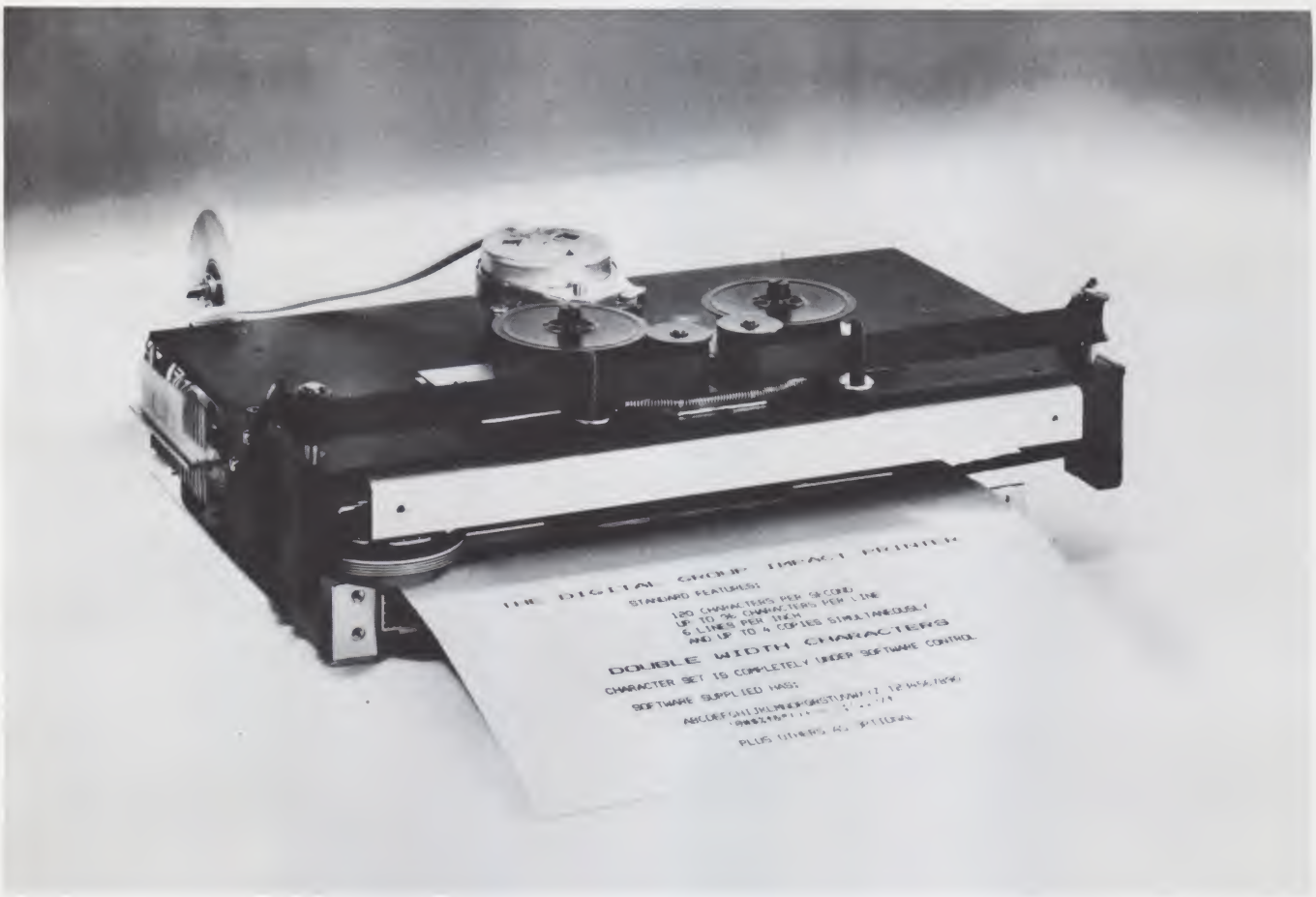
City State Zip N U

Credit Card Number

4 Numbers Above Name (MC) Good Thru

Signature

P-13



Print Your Heart Out.

With help from the Digital Group, naturally.

Now, that small computer system you own or have been considering for personal or business use suddenly becomes a lot more usable—with the addition of a full-size *impact* printer from the Digital Group. A printer designed for small computers that need big output (like yours).

With the Digital Group printer, you can print your heart out...and it won't cost an arm and a leg. The Digital Group printer is available for less than $500. That's right—$500.

Just look at these specifications:

- Fast—120 characters per second
- 96 characters per line
- 12 characters per inch horizontal
- 6 lines per inch
- Makes up to 4 copies simultaneously
- Character set and pitch variable under software control—double width characters, etc.
- 5 x 7 character matrix
- Ribbon has built-in re-inkers for a life of 10,000,000 characters
- Paper can be either a standard 8½-inch roll, fanfold or cut page
- Interfaces to 8-bit parallel ports

There are lots of capabilities and outstanding features of the Digital Group printer...and (as always) the best news is our price. Kit prices start as low as $495 for the printer and interface card. It simply can't be beat.

Find out all the facts about the Digital Group printer now. Just fill in the coupon below or give us a call for the details. We think you'll find a place for our printer in your system ...and in your heart.

---

the digital group

P.O. Box 6528
Denver, Colorado 80206
(303) 777-7133

Quick. I want to print my heart out.
Send me all the details on your full-size impact printer.

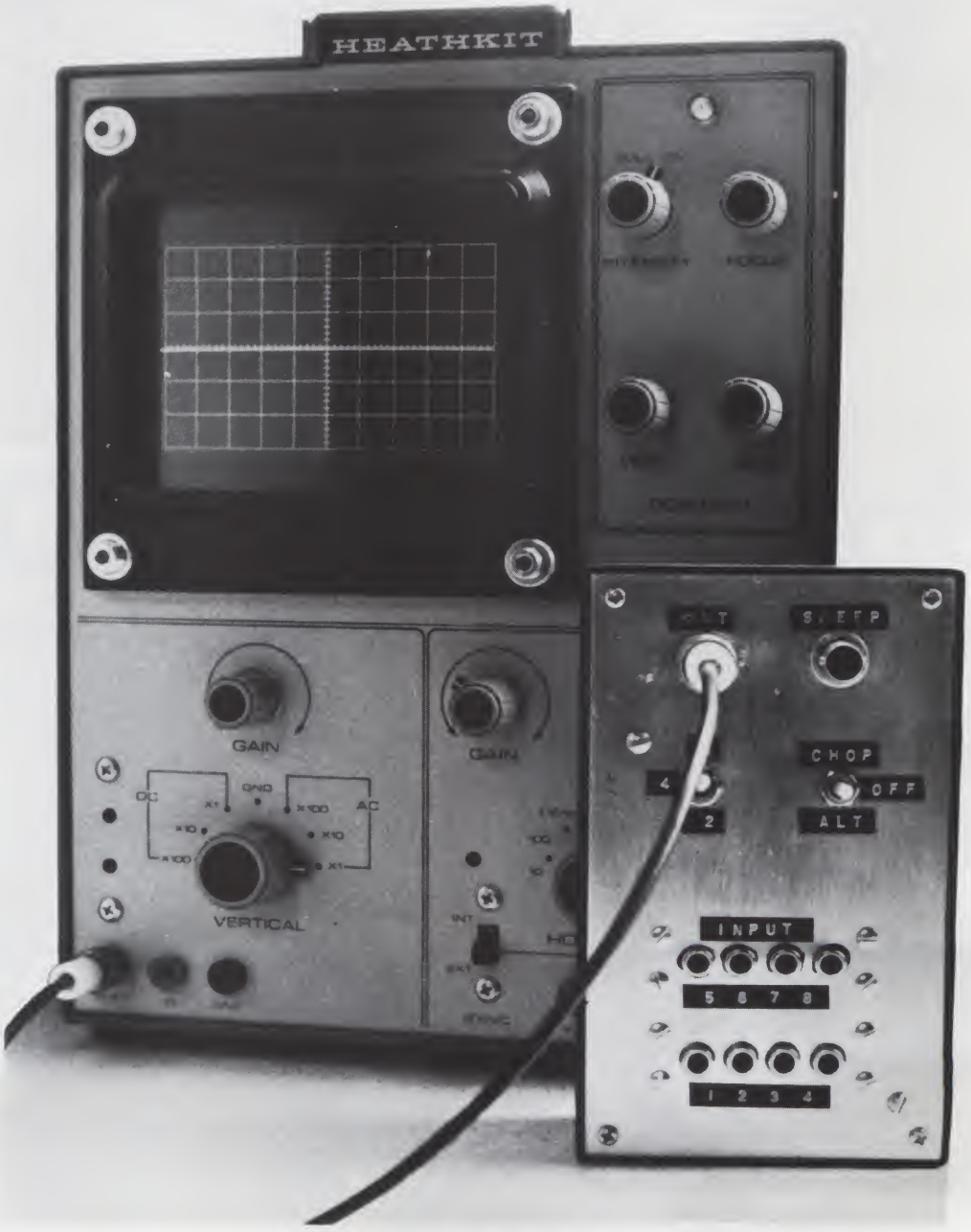
Name _____

Address _____

City/State/Zip _____

D-12 Please print.

Build an Eight Channel



The completed unit . . . ready to go to work.

W. J. Prudhomme
1405 Richland Ave.
Metairie LA 70001

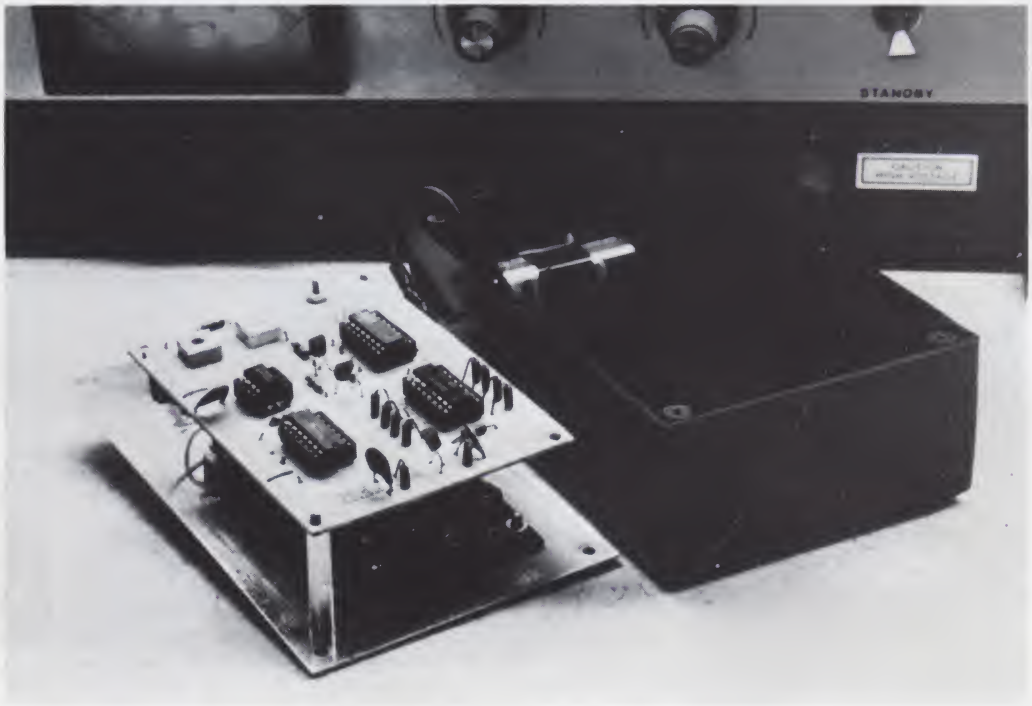
Multiplexer for Your Scope

If we put Bill's idea for an 8-trace adapter together with ''Nobody Knows The Troubles I've Seen'' from issue #1 (building the Logic Analyzer Box), plus a few more similar articles, before you know it we'll have a complement of computer test equipment we can all be proud of. Bill has another article coming describing an add-on triggered sweep for this unit. — John.

Troubleshooting digital circuits can sometimes be a hassle, since so many things are happening at various points in the circuit and usually at very high speeds. The situation is becoming so complex that design engineers and technicians are turning to logic analyzers to assist them in debugging these digital circuits. A logic analyzer can monitor various points in a circuit and display the logic states at these points including storage of the information in an internal memory.

This technique may work fine for digital design engineers who have access to thousands of dollars of test equipment, but what about the typical hobbyist on a limited budget? In my own case, I only have an FET voltmeter and an oscilloscope (single trace) which I use to trace logic levels. Many of you are probably using the instruments listed above in addition to a logic probe.

In most cases, these three instruments will be all you will ever need to debug digital equipment, although the process of finding the problem may be laborious. Why? Because all of the instruments mentioned above have one common disadvantage: each has the limited capability of displaying information on only one point in the circuit at a time. This can be time-consuming when there are many test points to be checked out.



Construction layout.

A better approach would be to monitor many key points in a circuit and to display this information simultaneously. This would allow you to check logic levels throughout the circuit and to see the timing relationships as they occur at the various test points. The question is, how do you monitor an entire digital circuit without spending a fortune on sophisticated test equipment?

Faced with this problem, I came across a solution recently that should be of interest to anyone seriously involved in digital electronics. This article describes the construction of a low cost (only 4 ICs are required) adapter that will convert any garden variety, single trace, oscilloscope into a professional grade eight channel scope. It accomplishes this by *multiplexing* eight input signals into one output for the scope's vertical amplifier. At the same time, discrete voltages are sequentially picked off of a resistor divider chain and added to the digital input signal. The net effect is that the single oscilloscope trace is sequentially shifted in discrete steps so fast that the image appears as eight individual traces. While the trace is being shifted each time, the digital signal information is being added to it so that the scope displays eight traces of digital information, all in sync.

How The Circuit Works

Referring to the block diagram shown in Fig. 1, the adapter uses a 74151 TTL digital multiplexer to sample eight individual signal inputs. The addressing for the 74151 requires 3 bits of a binary code to address all eight channels. As each channel is sampled, its logic level appears at pin 5 of the 74151 and is routed to the output connector of the adapter (which is connected to the vertical input of the scope). Note that these are digital level signals — waveshapes and input voltage levels are not preserved, only logical 1 or logical 0 states.

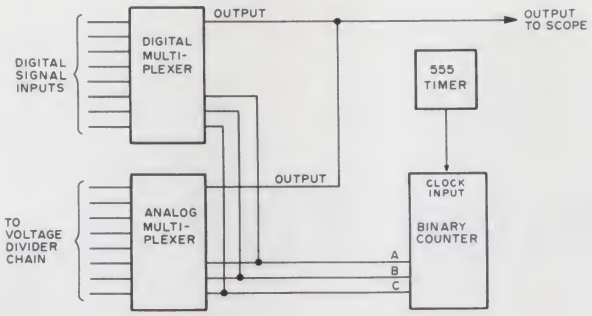


Fig. 1. Block diagram of the multiplexer showing the major components.

The CD4051 CMOS analog multiplexer is also connected to the same 3-bit address lines as the 74151 above. This means that it too is sequentially stepped through eight signal channels in unison with the 74151. The only difference is that the eight analog input signal lines are connected to a voltage divider consisting of nine 100 Ohm resistors. As the analog multiplexer sequentially samples each signal line, the analog voltage level appears at the output pin 3 of the CD4051 (see Fig. 2). This staircase voltage is summed through Q1 and added to the output line of the adapter, to be fed to the input of the scope's vertical amplifier.

Multiplexer front panel.

There are two modes of operation for the adapter: chopped and alternate sweep. In the *chop* mode, a 555 timer provides the clock rate for the 7493 binary counter. This counter provides the 3-bit binary code required to step the two multiplexer ICs through 2, 4 or 8 discrete steps. This is accomplished by selecting the proper address lines to provide a choice of either two, four or eight traces on the scope.

In the *alternate sweep* mode, the input of the 7493 is switched to a signal conditioning circuit which shapes the sweep signal from the scope and conditions it to be TTL compatible. The input resistor, R17, is sized to accept sweep voltages up to about 30 volts. For scopes with other sweep voltage levels, R17 may have to be redesigned.

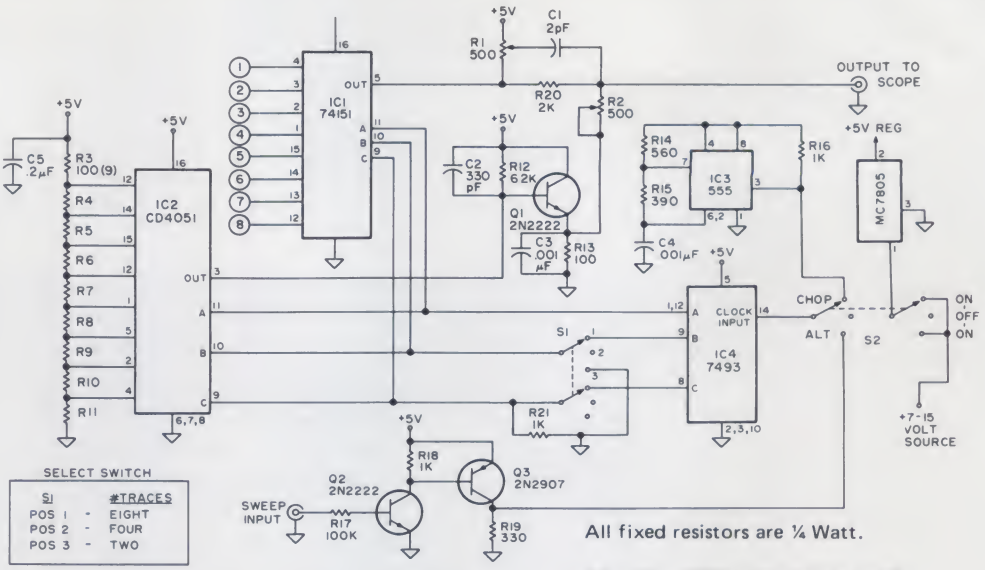
Three elements are included in the circuit for compensation. These elements include the R1/C1 combination, C2 and C5. You may want to experiment with different values of these components in order to get the cleanest set of multiple traces on your scope.

Capacitor C5 helps to produce a *flat* staircase pattern and its value should be adjusted for the best response. Also, capacitor C2 helps to eliminate overshoot at the end of each staircase step. The effect, if not compensated, will result in a thick trace line which is undesirable. The R1/C1 combination is also adjusted for a narrow trace on the scope.

To obtain the regulated +5 volts as required for the TTL elements, a MC7805 regulator was used. Since the total current drain of the adapter is about 60-70 mA, no heat sinking is required. Consequently, the regulator can be mounted directly on the circuit board. To function properly, at least 7 volts must be applied from the battery pack. To meet this require-

ment, five 1.5 volt batteries in series or a 9 volt transistor battery may be used. However, since current draw is about 60-70 mA, I would recommend rechargeable batteries or an ac supply. I am working on a new *all CMOS* design to reduce power consumption and will discuss this in more detail later in the article.

One final word about the 74151 8-channel multiplexer. As mentioned earlier, it is the nature of this device not to preserve input signal analog voltage levels — only digital logic levels. However, since the purpose of this adapter is to troubleshoot only digital circuits, this characteristic should not affect the adapter's usefulness. After all, we are mainly interested in the timing relationship of various digital logic levels, and not necessarily their actual analog value. So in this type of application, the 8-channel multiplexer described here should prove very useful in saving you time and frustration in the debugging of your next digital project.



All fixed resistors are ¼ Watt.

S1, S2 DPDT miniature with center off position.

Fig. 2. Schematic diagram for the adapter.

Construction Details

The multiplexer adapter will fit neatly in a 6¼" x 3¾" x 2" plastic instrument case. I used BNC connectors for the scope output and sweep input to match existing cables. However, since these types of connectors are relatively expensive, I switched to standard RCA phono sockets for the 8-channel signal input connections. This decision was made primarily to keep the overall cost of the unit within reason. These sockets work fine, although your own personal preferences will govern your choice here and the type is not critical.

The circuit board layout is shown in Fig. 3 with parts placement indicated in Fig. 4. The actual circuit layout is not critical, although I would recommend sockets for the ICs and careful handling of the CD4051.

Miniature DPDT switches (the on-off-on type) were used both for the mode select/power and channel selection functions. The use of these special DPDT switches with a pull-up resistor for the *off* position eliminates the need for a three position rotary switch to select 2, 4, or 8-channel operation. Also, the addressing lines are connected so that the traces are always centered on the screen for either 2, 4, or 8-channel selection.

Conclusion

The multiplexer adapter performs well in displaying multiple scope traces for debugging digital circuits. It works even better if it is used with a scope having a trig-

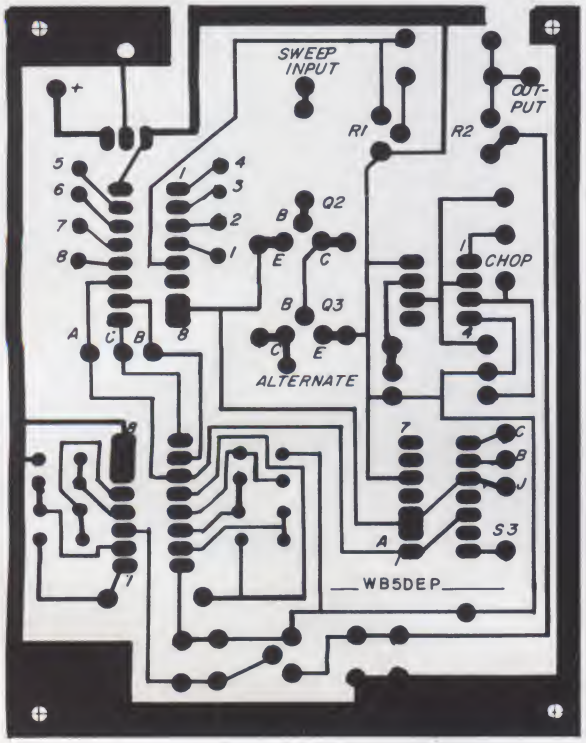


Fig. 3. PC board layout (full size).

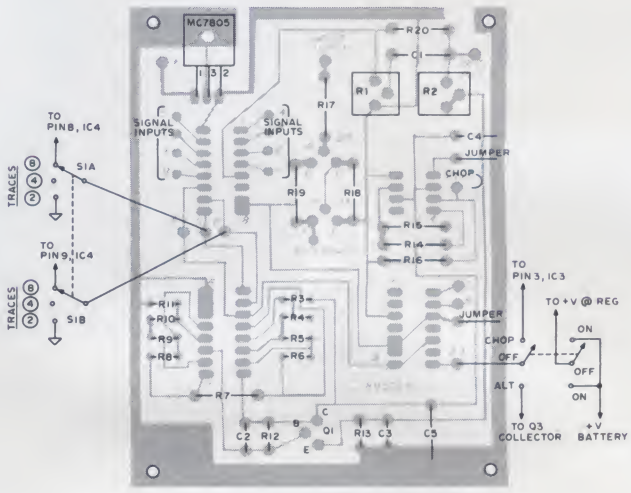


Fig. 4. Component placement on the circuit board.

gered sweep feature. The reason for this is that a triggered sweep scope will allow you to lock the trace on an input signal, rather than trying to lock on the chopped frequency generated by the 555 timer.

If your scope does not have an externally triggered sweep, the adapter will still provide satisfactory performance. It is just that a triggered sweep makes it easier to pick out the signal you want to lock into.

Also, with a 60-70 mA current drain, battery life may be short depending on the size of the batteries chosen for your instrument. As mentioned earlier, either rechargeable nicads, and ac power supply or deriving power from the circuit being tested should be considered. If the latter approach is taken and the supply voltage is +5 volts, it will be necessary to take the MC7805 regulator out of the adapter's circuit. The reason being that the MC7805 requires an input voltagy of at least +7 volts to function properly.

To reduce power consumption, I am presently designing an *all CMOS* version and if there's enough interest, I may have a follow-up article describing the results. At present, I am considering another CD4051 (with adequate input conditioning to replace the 74151 in the existing design. Also, a CD4029 binary counter to replace the TTL 7493 counter. To complete the design, perhaps a CD4047 low power multivibrator could replace the 555 timer.

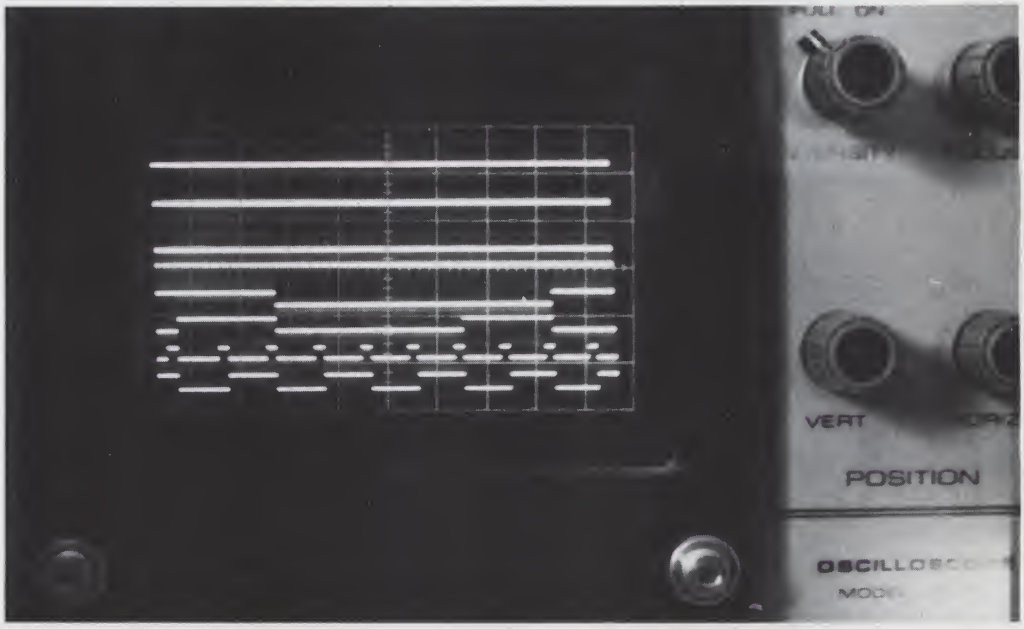
The primary advantages of going to CMOS devices are:

1. Low power consumption (a 9 volt battery would last over a year).
2. TTL or CMOS logic levels could be displayed.
3. The MC7805 voltage regulator could be eliminated.

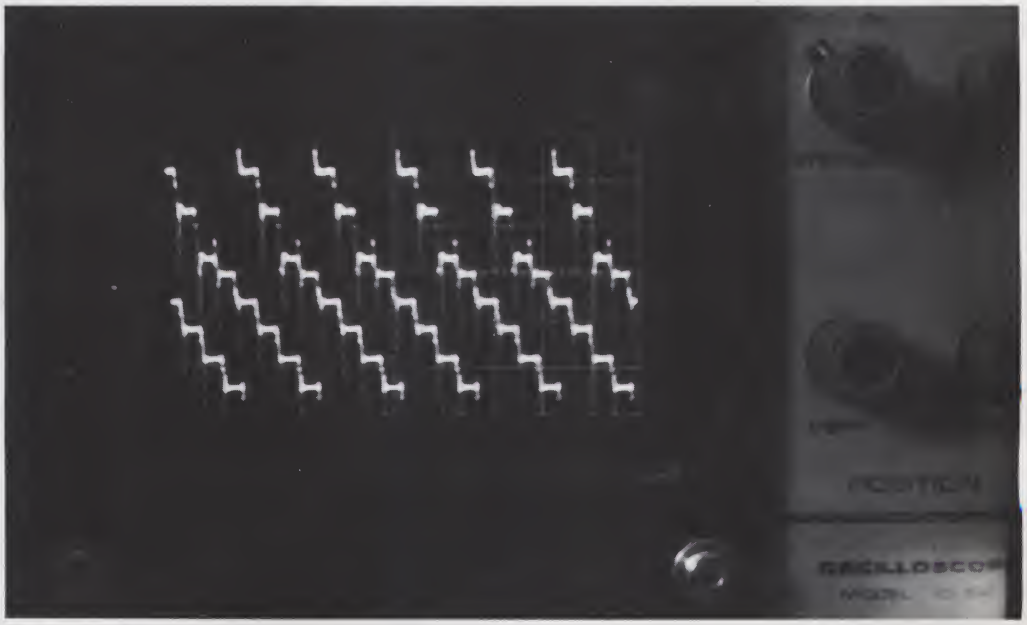
I'm firmly convinced that CMOS devices are the way to go and it's also my personal opinion that eventually CMOS logic elements will outnumber TTL logic devices in a multitude of applications. I feel safe with that assumption even though many hobbyists have voiced objections to CMOS susceptibility to static voltage damage. Actually, I have found them to be very rugged and I have never experienced a CMOS device failure due to handling or static electricity. In fact, once you get used to them, I'm sure you will agree that CMOS devices offer many advantages not found in the TTL family of logic devices. ■

Reference

"Chopping Mode Improves Multiple-trace Display," *Electronic Magazine*, October 1976.



With the multiplexer in the 8-channel chopped mode, eight traces are shown here with digital information on the four bottom lines. The signal sources are various pins on a 7490 counter.



In this photo, all eight channel inputs were disconnected from signal sources and the oscilloscope was sync'd to the output of the multiplexer. This was done to illustrate the stepped waveform as the multiplexer switches to each signal line.

Note the difference in levels between the second and third traces from the top as compared to the other steps. This was caused by not using precision resistors in the resistor divider chain connected to the CD4051. This effect can be overcome by matching each resistor used in the divider chain before installation.

Also note the overshoot at each step that gives the effect of a slight halo over each trace in the other photos. This effect was not found to be objectionable in viewing the digital traces.

Experience the excitement of owning the world's finest *personal* computer – IMSAI 8080

Waiting for you — all the incredible performance and power of the IMSAI 8080. And at a price you would normally pay for a fine home music system.

Introduced less than two years ago, the IMSAI 8080 is sold worldwide and acknowledged as the finest *personal* computer available.

WORLD OF USES

The IMSAI 8080 is a superbly engineered, quality computer. It is versatile, expandable and powerful, putting literally hundreds of applications and uses at your fingertips. Imagine sitting at your desk and enjoying interaction with your IMSAI 8080! Press the *on* switch and you're ready for game playing, research, education, business applications, or for laboratory instrument control. It has all the power you need, and more, to make your application ideas come alive.

GROWS WITH YOU

The IMSAI 8080 is designed for many years of pleasure. With its open-chassis engineering you can expand your system by adding peripherals and interfaces. The 22-slots and 28 amp power supply mean that you can plug-in today's, plus *tomorrow's* modules.

Right now you can add a module for displaying color graphics and characters on TV; a ready-to-use keyboard; small and large printers, and a single interface that lets you attach multiple devices including a cassette tape recorder. Expect the latest, exciting equipment from IMSAI. We are committed to leadership in this expanding technology.

EASY TO PROGRAM

With our BASIC language you can operate the IMSAI 8080 quickly and easily. Technically knowledgeable? Use our assembly language to develop sophisticated and unique software.

If you're thinking personal computer, treat yourself to the very best – IMSAI 8080

Send for free four-color brochure or $1 for catalog. Call or write for name of nearest dealer.

IMSAI

IMSAI Manufacturing Corporation
14860 Wicks Blvd.
San Leandro, CA 94577
(415) 483-2093
TWX 910-366 7287

European Distributor
Harper Industry Products, Ltd
6079 Sprendlingen 2
West Germany

Sorting Routines

... explanation of common sorting techniques

You're going to appreciate (and need) the sorting routines Andrew has presented here regardless of whether you're developing game programs, educational, home or small business applications software, or anything else. Not only does he discuss three of the most popular sorting techniques (in BASIC) but he also has a tip on converting a BASIC program using SWAP statements to one which doesn't. — John.

Andrew J. Rerko
R.D. #2 Box 222
Smithfield PA 15478

After reading the first issue of *Kilobaud*, I decided that this was the kind of magazine for which I would try writing an article. Trying to think of a topic, I found it in the program written by George L. Haller in his article, "Computers in Golf" (*Kilobaud #1,* pages 96-98).

I will discuss the sort routine of lines 590 thru 650. First of all, I believe the less-than sign (<) was left out by accident in line 610. Also, the BASIC that I am using does not have the SWAP statement (Line 620). Therefore,

```
SWAP A(I), A(I+1)
```

will be replaced by,

```
T = A(I)
A(I) = A(I+1)
A(I+1) = T
```

What is really happening in these three lines is quite simple. First, the value in A(I) is placed temporarily in T, then the value in A(I+1) is placed in A(I), and last, the value in T is placed in A(I+1); completing the SWAP.

This now leads to my Ripple Sorting Routine, which is almost identical to George L. Haller's except that it is in general form. The Ripple Sort is shown in Program A.

In all the routines to be discussed, N will represent the number of items to be sorted in array A. The program is comparing the first item with the second and if they are not in order, they are switched. Then the second item is compared with the third, and so on. When it reaches the end of the list, it will start over and repeat this process until there are no changes being made.

Examine the program line by line. Line 10 is setting up a check, C, and sets it to zero. Line 20 sets up a loop from 1 to the next to last number in the list. In line 30 the I^{th} position is compared with the $(I+1)^{th}$ position. This is the reason that the loop goes to N-1; if it were allowed to go to N, statement 30 would compare the N^{th} and $(N+1)^{th}$ positions which would be outside the list to be sorted. If the two numbers being compared are in the correct order, branch to line 80. If the numbers are in the wrong order, they are switched by lines 40, 50, and 60 which is the SWAP routine explained above. Line 70 sets the check to 1, indicating that a SWAP has occurred. Line 80 is the end of the loop, which will be repeated N-1 times as indicated in line 20. Line 90 checks to see if there were any changes made during the loop. If there were, then there *may* still be more necessary. Therefore, the program branches back to line 10 which repeats the process until the loop is completed without a swap being made. The check, C, will be zero and the list will be in order.

Now refer to Table 1 for a look at an array of four numbers and their order after each passes thru the loop. (I have indicated the numbers being compared/swapped by an underline. Compare a column with the previous column to see if a change was made.) Note that in the third execution of the loop, in the first pass, the list is in order. But, there was a swap made during the third execution and therefore the check, C, is 1; and the loop must be executed a fourth time. Since there will be no swaps made during the fourth execution, the list is known to be in order.

Take a look at the number of comparisons being made. First notice that it executes the loop four times, which in general form is N executions. It also makes three comparisons during each execution which is N-1 comparisons. Therefore, it makes N(N-1) or N^2-N comparisons. Realize though, that this is the worst possible case. If the list were partially in the correct order, this number would be smaller. If the list were in perfect order to begin with, there would be N-1 comparisons. The process used in this program can be shortened. Note the numbers indicated by the asterisks(*). They do not change, yet they are compared nonetheless. You realize, of course, that this list has only four items, imagine if it consisted of 50 items. We can eliminate most of these unnecessary comparisons by the inclusion of just two statements. A listing of the modified version is shown in Program B. Each

time line 12 is executed it decreases the number of items to be compared. Line 15 is necessary to prevent line 20 from being executed with N being zero, which may cause an error, and it also prevents the program from executing the loop one extra, unnecessary time. Also note line 20 has been altered due to line 12. You can see the need for this. Refer to Table 2 for the same group of numbers being sorted with this program. After the third execution, N will be zero and therefore a branch to line 100 will occur, which ends the routine.

Now look at the number of comparisons in the worst case of this program. First, there will be only three executions of the loop, which in general terms is N-1 executions. During each execution of the loop the number of passes and, therefore, comparisons decreases. The number of comparisons will be as $(N^2+N)/2$ (see Table 3).

Take a look at Table 4 to appreciate a few of the savings in number of comparisons of the second program over the first.

You can see that as the number of items to be sorted increases so does the savings. I must again *stress* that these are worst case examples. If the list were partially in order, the number of comparisons could drop drastically. If the list were completely in order to begin with, in either program, there would be N-1 comparisons being made. Two other methods for sorting a list of numbers will also be discussed. One of these is the Bubble Sort. Program C is a listing of same. Instead of going thru this program line by line, let me just explain what is happening. When I = 1, J goes from 2 thru N, which compares the first number with all the rest. If at any time one of these numbers is smaller than the first number, they will be swapped. After the J loop is completed, the smallest number of the array will be in the first position indicated by I = 1. Then I = 2 and J goes from 3 thru N, which now compares the second number with all the rest. When the J loop is completed, the second smallest number will be in the second position. This process will continue until the I loop is completed. The list will be in order. Refer to Table 5 for the same list of numbers to be sorted using this program.

Looking at the number of comparisons necessary, you will see that it follows the same pattern as outlined in Table 3:

N-1, N-2, N-3 . . . N-(N-1)

which results in N(N-1)/2 or $(N^2-N)/2$ comparisons, which will hold for all cases. There is no worst case or best case. Even if the list was in perfect order, it would take $(N^2-N)/2$ comparisons.

One more type of sorting deserves to be looked at: the Ranking Sort. It is in a separate class in that another array of memory is necessary. The original list is left untouched in array A and the sorted list will be in array B. An example of the Ranking Sort is shown in Program D.

This routine is valuable for the following reason. Suppose you are inputting a list of numbers that are to be sorted. For all of the other programs the computer must sit and wait until the complete list is inputted before it can begin to sort them. With

```
10 C = 0
20 FOR I = 1 TO N-1
30 IF A(I) < = A(I+1) THEN 80
40 T = A(I)
50 A(I) = A(I+1)
60 A(I+1) = T
70 C = 1
80 NEXT I
90 IF C = 1 THEN 10
```

Program A. Ripple Sort

```
10  C = 0
12  N = N - 1
15  IF N = 0 THEN 100
20  FOR I = 1 TO N
30  IF A(I) <= A(I+1) THEN 80
40  T = A(I)
50  A(I) = A(I+1)
60  A(I+1) = T
70  C = 1
80  NEXT I
90  IF C = 1 THEN 10
100 (next line after sort routine)
```

Program B. Modified Ripple Sort

```
10 FOR I = 1 TO N-1
20 FOR J = I+1 TO N
30 IF A(I) <= A(J) THEN 70
40 T = A(I)
50 A(I) = A(J)
60 A(J) = T
70 NEXT J
80 NEXT I
```

Program C. Bubble Sort

```
10 B(1) = A(1)
20 FOR I = 2 TO N
30 FOR J = I TO 2 STEP -1
40 IF B(J-1) <= A(I) THEN 80
50 B(J) = B(J-1)
60 NEXT J
70 J = J - 1
80 B(J) = A(I)
90 NEXT I
```

Program D. Ranking Sort

Array	Orig. List	First Execution of Loop			Second Execution of Loop		
		1st Pass	2nd Pass	3rd Pass	1st Pass	2nd Pass	3rd Pass
A(1)=	20	19	19	19	18	18	18
A(2)=	19	20	18	18	19	17	17
A(3)=	18	18	20	17	17	19	19*
A(4)=	17	17	17	20	20*	20*	20*

	Third Execution of Loop			Fourth Execution of Loop		
	1st Pass	2nd Pass	3rd Pass	1st Pass	2nd Pass	3rd Pass
	17	17*	17*	17*	17*	17*
	18	18*	18*	18*	18*	18*
	19*	19*	19*	19*	19*	19*
	20*	20*	20*	20*	20*	20*

(Notice the pattern formed by the underlines)

Table 1. Execution sequence (trace) for the Ripple Sort.

First Execution N-1 Passes; Second Exec. N-2 Passes; Third Exec. N-3 Passes; . . . (N-1)th Exec. N-(N-1) Passes

From Algebra, the total number of comparisons is:

N-1 + N-2 + N-3 + . . . + 1.

But also remember statement 15, which is a comparison (that was not in the original program), will be executed N times. This must also be added to the above sum, giving:

N + N-1 + N-2 + N-3 + . . . + 1.

Again from Algebra we get N(N+1)/2 or $(N^2+N)/2$ comparisons.

Table 3. General statement of Ripple Sort execution.

Number of Numbers Sorted	Number of Comparisons First Routine	Second Routine
N	$N^2 - N$	$(N^2 + N)/2$
4	12	10
6	30	21
10	90	55
50	2450	1275
100	9900	5050

Table 4. Advantages of modified Ripple Sort.

INCREASING ORDER: the first item in a list is smaller than the second, and the second is smaller than the third, etc.
Ex: 10, 20, 30, 40, 50, . . .
NONDECREASING ORDER: the first item in a list is not larger than the second, and the second is not larger than the third, etc.
Ex. 10, 20, 20, 30, 30, 30, 40, . . .
DECREASING ORDER: the first item in a list is larger than the second, and the second is larger than the third, etc.
Ex. 60, 50, 40, 30, 20, . . .
NONINCREASING ORDER: the first item in a list is not smaller than the second, and the second is not smaller than the third, etc.
Ex. 60, 50, 50, 40, 30, 30, 10, . . .

Table 7. Glossary of Terms.

Array	Orig. List	I = 1 J=2	J=3	J=4	I = 2 J=3	J=4	I = 3 J=4
A(1)=	20						
A(2)=	19						
A(3)=	18	19	18	17	17	17	17
A(4)=	17	20	20	20	19	18	18
		18	19	19	20	20	19
		17	17	18	18	19	20

(Again, note the pattern formed by the underlines.)

Table 5. Execution sequence for the Bubble Sort.

1st Array	Orig. List	2nd Array	Stmt	I=2 J=2	I=3 J=3	J=2	J=4	I=4 J=3	J=2
A(1)=	20	B(1)=	10	19	19	18	18	18	17
A(2)=	19	B(2)=	20	10		19	19		18
A(3)=	18	B(3)=			20	20		19	19
A(4)=	17	B(4)=					20	20	20

Table 6. Execution sequence for the Ranking Sort.

Array	Orig. List	First Execution of Loop 1st Pass	2nd Pass	3rd Pass	Second Execution of Loop 1st Pass	2nd Pass	Third Execution of Loop 1st Pass
A(1)=	20						
A(2)=	19	19	19	19	18	18	17
A(3)=	18	20	18	18	19	17	18
A(4)=	17	18	20	17	17	19	19
		17	17	20	20	20	20

Table 2. Execution sequence for modified Ripple Sort.

this routine that is not true. After each input, by putting an input routine between lines 20 and 30, the program will place that value in its proper place in array B. At any time you may stop the program and the numbers inputted so far would be in the correct order in array B.

The best way to see what is happening here is to follow through with a few steps. The first number in array A is placed in array B. Then A(2) is compared with B(1). If A(2) is larger, it is placed directly in B(2). If it is smaller than B(1), B(1) is moved to B(2), and A(2) is placed in B(1). Either way, the list in B is in order. Then, A(3) is compared with B(2). If A(3) is larger than B(2), it is placed directly in B(3). If A(3) is smaller than B(2), B(2) is moved to B(3). Then A(3) is compared with B(1). If A(3) is larger than B(1), A(3) is placed in the *empty* slot, B(2). If A(3) is smaller than B(1), B(1) is moved to B(2) and A(3) is put in B(1). Again, the list in B is in order. This process will be repeated with each number of the original list. If you have difficulty in following this, take pencil and paper and try working it through as outlined in this paragraph. You will see the beauty of it.

See Table 6 for a trace of the same list of numbers being sorted with this program (which, incidentally, is very difficult to show). Note that the number of comparisons increases during each execution of the I loop. It follows the pattern,

1, 2, 3, . . . N-1

which is equivalent to N(N-1)/2 or $(N^2-N)/2$ comparisons. This is a worst case example. It would be smaller as the list is more in order. If the list were in perfect order, there would be N-1 comparisons made.

All of the above routines sort in a nondecreasing (increasing) order. This could easily be changed into nonincreasing (decreasing) order by changing the less-than sign (<) to a greater-than sign (>). The above routines could also be used to sort alphabetic information by changing the appropriate variables to string variables. ■

EVERY computer hobbyist should subscribe to 73

How does over 300 pages of articles on hobby computers published in 1976 grab you? ■ ■ ■ how many other magazines can claim that much? It may come as a big surprise to you, but there are a lot of non-ham readers of 73 ■ ■ ■ 73's barrage of computer articles has not entirely escaped the eyes of all computerists.

YES, 73 certainly does cover the 25 or so hobbies which are classed together as amateur radio ■ ■ ■ radioteletype (RTTY), slow scan television (SSTV), DXing (contacting foreign countries), FAX (facimile transmission), FM and repeaters, satellite use, moonbouncing, contests, experimenting, home construction, and even rag chewing. There are articles on building and using test equipment, on timers, weather satellite systems, wind speed measuring, and hundreds of other related and unrelated subjects.

73 is the LARGEST of the ham magazines ■ ■ ■ the January 1977 issue ran 50 articles and was over 200 pages. Add to all of that the irreverence of Wayne Green and his editorials and you have a magazine that most readers read from cover to cover every month ■ ■ ■ NO MATTER HOW MANY OTHER MAGAZINES THEY GET.

Do you really want to continue to miss all those computer articles?

SUBSCRIPTION TO 73 MAGAZINE

☐ $15 One year* ☐ $36 Three years* ☐ $150 Life

Name _____ Call (if any) _____

Address _____

City _____ State _____ Zip _____

*U.S. and Canada ONLY — Please write for foreign rates
Allow 6-8 weeks for subscription processing.

☐ Cash enclosed ☐ Check enclosed ☐ Money order enclosed

Charge to:

☐ Master Charge ☐ BankAmericard ☐ American Express

Credit Card # _____ Interbank # _____

Expiration date _____ Signature _____

☐ Bill me direct Signature _____

YOU MAY USE THE CARD INSIDE THE BACK COVER FOR THIS ORDER.

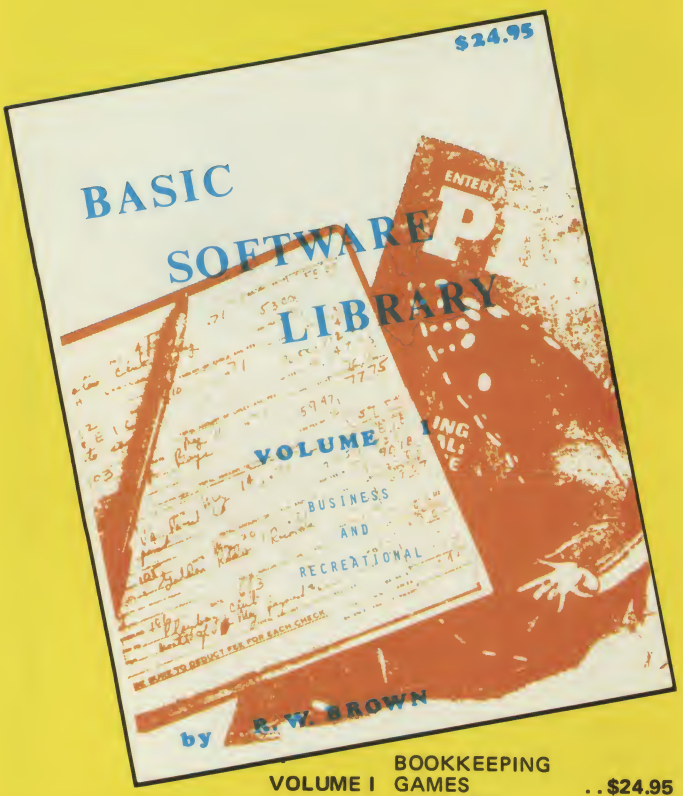
73 MAGAZINE, PETERBOROUGH NH 03458 USA

Toll Free subscription number 800-258-5473 K4/77

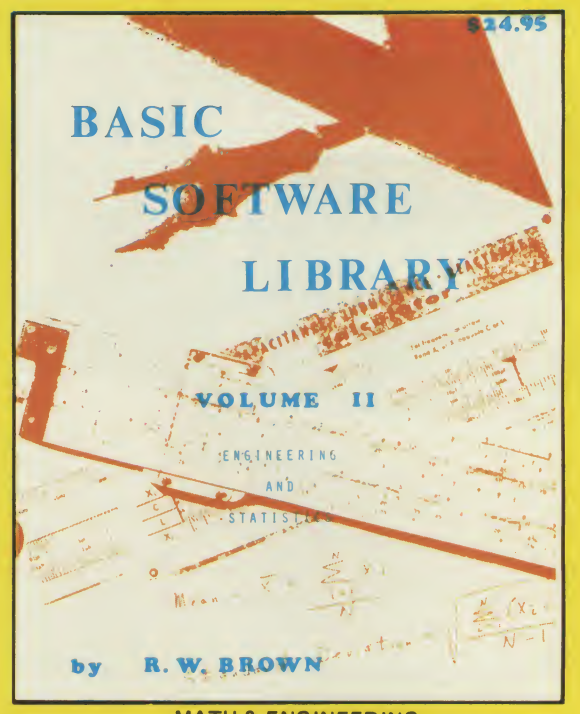
The BASIC SOFT

The "HOW-TO-DO IT" Books for the "DO-IT" Person

Written in compatible BASIC immediately executable in ANY computer with at least 4K, NO other peripherals needed.



VOLUME I BOOKKEEPING
GAMES
PICTURES
..$24.95



VOLUME II MATH & ENGINEERING
PLOTTING & STAT
BASIC STATEMENT DEF
..$24.95

ARIZONA
Byte Shop of Phoenix
Tempe, AZ 85281
(602) 894-1129

Byte Shop of Phoenix-West
Phoenix, AZ 80529
/602) 942-7300

Desert Data Computer Store
Tucson, AZ 85702

CALIFORNIA
A-VID Electronics
Long Beach, CA 90806
(213) 426-5526

Byte Shop of Berkeley
Berkeley, CA 94703
(415) 845-6366

Byte Shop of Campbell
San Jose, CA 95124
(408) 377-4685

Byte Shop of Diablo Valley
Walnut Creek, CA 94596
(415) 933-6252

Byte Shop of Fresno
Fresno, CA 93703

Byte Shop of Hayward
Hayward, CA 94541
(415) 537-BYTE

Byte Shop of Lawndale
Lawndale, CA 90260
(213) 371-2421

Byte Shop of Mt. View
Mt. View, CA 94040
(415) 969-5464

Byte Shop of Palo Alto
Palo Alto, CA 94306
(415) 327-8080

Byte Shop of Pasadena
Pasadena, CA 91101
(213) 684-3311

Byte Shop of Sacramento
Citrus Heights, CA 95610
(916) 726-2557

Byte Shop of San Diego
San Diego, CA 92111
(714) 565-8008

Byte Shop of the San
Fernando Valley
Tarzana, CA 93156
(213) 343-3919

Byte Shop of San Jose
San Jose, CA 95123
(408) 226-8383

Byte Shop of San Mateo
San Mateo, CA 94403
(415) 341-4200

Byte Shop of San Rafael
San Rafael, CA 94901
(415) 457-9311

Byte Shop of Santa Barbara
Santa Barbara, CA 93101
(805) 966-2557

Byte Shop of Santa Clara
Santa Clara, CA 95051
(408) 249-4221

Byte Shop/Thousand Oaks
Thousand Oaks, CA 91360
(805) 497-9595

Byte Shop of Westminster
Westminster, CA 92683
(714) 894-9131

Byte Shops, Inc.
Sunnyvale, CA 94086
(408) 734-9000

The Computer Mart
Orange, CA 92667
(714) 633-1222

The Computer Store
Santa Monica, CA 90401
(213) 451-0713

People's Computer Shop
Sherman Oaks, CA 91423
(213) 789-7514

CANADA
Byte Shop of Vancouver
Vancouver 9, B.C.
(604) 736-7221

The Pacific Computer Store
Vancouver, B.C. V5R 2J4
(604) 438-DATA

Trintronics
Toronto, Ontario
(416) 598-0262

COLORADO
Byte Shop/Arapahoe Co.
Englewood, CO 80110
(303) 761-6232

Byte Shop of Boulder
Boulder, CO 80301
(303) 449-6233

FLORIDA
Byte Shop of Cocoa Bch.
Cocoa Beach, FL 32931
(305) 784-1881

Byte Shop of Miami
Miami, FL 33155
(305) 264-BYTE

Computer Hut
Miami Lakes, FL 33014
(305) 821-2667

MicroComputer Systems Inc.,
Tampa, FL 33609
(813) 879-4301

ILLINOIS
The Data Domain
Evanston, IL 60201
(312) 328-6800

INDIANA
The Data Domain
Bloomington, IN 47401
(812) 334-3607

The Home Computer Store
Indianapolis, IN 46229
(317) 894-3319

JAPAN
Byte Shop of Tokyo
2-9-9 Sotokanda
Chiyodaku, Tokyo
Kiyotake Ikeda

KENTUCKY
The Data Domain
Lexington, KY 40502
(606) 233-3346

MARYLAND
Computer Workshop
Rockville, MD 20852
(301) 468-0455

MINNESOTA
Byte Shop of Eagan
Eagan, MN 55121
(612) 452-1841

MISSOURI
Computer Systems Center
of St. Louis, Inc.
Chesterfield, MO 63017
(314) 576-5020

Computer Workshop
Kansas City, MO 64152
(816) 741-5055

NEW HAMPSHIRE
Computer Mart of NH
Nashua, NH 03060
(603) 883-2386

Microcomputers, Inc.
Nashua, NH 03060
(603) 889-1646

NEW JERSEY
Hoboken Computer Works
Hoboken, NJ 07030
(201) 420-1644

NEW YORK
Byte Shop of Levittown
Levittown, NY 11756
(516) 731-8116

Computer Mart of Long I
East Meadow, NY 11554
(516) 794-0510

The Computer Mart of NY
New York, NY 10001
(212) 279-1048

Synchro-Sound Enterprises
Hollis, NY 11423
(212) 468-7067

OKLAHOMA
High Technology
Oklahoma City, OK 73116
(405) 842-2021

OREGON
Byte Shop of Beaverton
Beaverton, OR 97005
(503) 644-2686

Byte Shop of Portland
Portland, OR 97201
(503) 223-3496

PENNSYLVANIA
Byte Shop of Bryn Mawr
Bryn Mawr, PA 19010
(215) 525-7712

Personal Computer Corp.
Frazer, PA 19355
(215) 647-8460

RHODE ISLAND
Computer Power Inc.
Warwick, RI 02886
(401) 738-4477

SOUTH CAROLINA
Byte Shop of Columbia
Columbia, SC 29205
(803) 771-7824

TENNESSEE
Micro Computer Systems
Knoxville, TN 37922
(615) 966-9849

UTAH
Byte Shop/Salt Lake City
Salt Lake City, UT 84111

VIRGINIA
Shire Enterprises
Richmond, VA 23222
(804) 321-4560

WISCONSIN
The Milwaukee Computer Store
Milwaukee, WI 53213
(414) 259-9140

Available NOW ! ! !

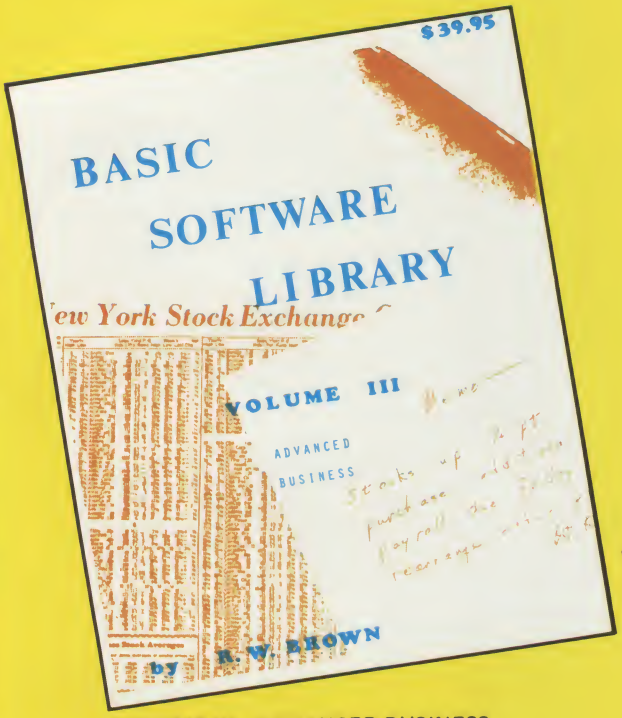
at most

computer stores

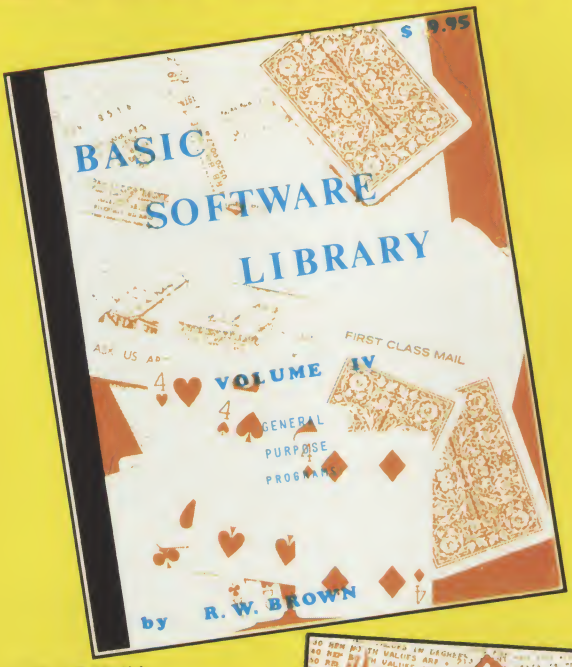
STOP BY OUR BOOTH (e615) AT THE SAN

WARE LIBRARY

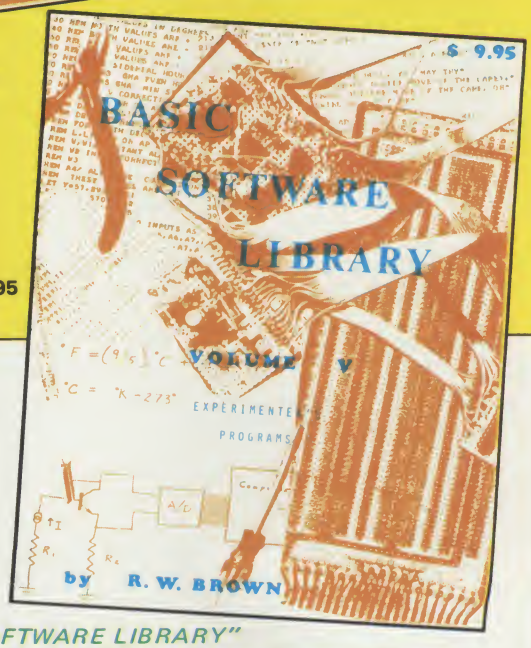
This LIBRARY is a complete do it yourself kit.
Knowledge of programming not required.
EASY to read and USE.



VOLUME III ADVANCED BUSINESS
BILLING
INVENTORY . . $39.95
PAYROLL



VOLUME IV
GENERAL PURPOSE . . $9.95



VOLUME V
EXPERIMENTER . . $9.95

FUTURE ADDITION TO THE "BASIC SOFTWARE LIBRARY"
Volume VI (A Complete Business System — $49.95) General Ledger System — Taxes, Pyrl, W-2's, Inventory, Depr., Financial Statements, etc. AVAILABLE MID SUMMER

SCIENTIFIC RESEARCH

1712-K FARMINGTON COURT
CROFTON MD 21114
Phone Orders call (800) 638-9194
Information and Maryland Residents Call (301)-721-1148

Add $1.50 per volume for postage and handling.

FRANCISCO COMPUTER FAIRE and say "HI!"

Number Rounding

Jack A. Inman
922 E. Edgecomb Avenue
Covina CA 91724

The high precision accuracy arithmetic which most of us have available to us in the various versions of BASIC is great. There are times when we would like to have a final printout with only the most significant decimal places shown. One instance would be when we are working with dollars and cents. In this instance we are not used to seeing up to nine decimal places or even scientific notation for the figures.

This little article hopefully will assist you in creating printouts which will be more and more meaningful. It is written to show one formula that will round out the decimal places in your printout for you. I do not claim originality for the formula as I found it in the book *BASIC* written by Robert L. Albrecht, LeRoy Finkle, and Jerald R. Brown. (This book is available from *Kilobaud Magazine*.)

What I did was write a program to show the formula and how it works in a step by step printout of the mathematical operations. Fig. 1 is a complete listing of the program which I included for those who might like to see how I created the printout for the run found in Fig. 2.

The program consists mostly of PRINT statements. There are a few directing statements such as those found in lines 260 and 380. Mathematical computations are made in the program as required to show what the computer does in each step toward the final result. Some very simple control computations are made and these are used to control the progress of the program.

The program was written in 4K BASIC from Southwest Technical Products Corporation for use in their M6800 computer.

The formula described in the program can be used in most versions of BASIC.

```
0001 REM **** NUMBER ROUNDING ROUTINE **********
0002 REM **** PROGRAM BY JACK INMAN ************
0003 REM **** DECEMBER 1976 ********************
0005 PRINT
0006 PRINT TAB(15);"NUMBER ROUNDING ROUTINE"
0007 PRINT TAB(18);"WRITTEN IN BASIC"
0008 PRINT
0010 PRINT "HERE IS A FORMULA WHICH IS VALUABLE IF YOU WANT TO"
0020 PRINT "ROUND OUT A NUMBER IN A PROGRAM. THE FORMULA"
0030 PRINT "IS AS FOLLOWS;"
0040 PRINT
0050 PRINT TAB(10);"LET X=INT(N*100+.5)/100"
0060 PRINT
0070 PRINT "EVEN MORE COMPLEX FORMULAS CAN BE ROUNDED SUCH AS;"
0071 PRINT
0075 PRINT TAB(10);"LET X=INT(N*N/N*100+.5)/100"
0076 PRINT
0077 PRINT "THE VARIABLE N IN THE ABOVE CAN BE ANY NUMBER OR"
0078 PRINT "VARIABLE YOU CHOOSE TO USE."
0079 PRINT
0080 PRINT "TO SEE HOW IT WORKS THE COMPUTER WILL ASK FOR A NUMBER"
0085 PRINT "TO BE ROUNDED. THEN IT WILL ROUND THE NUMBER EXPLAINING"
0086 PRINT "EACH STEP AS IT GOES THROUGH IT. HERE WE GO."
0090 PRINT
0095 PRINT
0100 PRINT "WE WILL USE X AS OUR VARIABLE IN THE RUN"
0105 PRINT
0106 LET Y=Y+1
0110 PRINT "INPUT THE NUMBER YOU WANT ROUNDED OUT          ";
0120 INPUT X
0130 PRINT
0140 PRINT "FIRST THE COMPUTER MULTIPLIES X BY 100"
0150 PRINT "THIS SHIFTS THE NUMBER TWO DECIMAL PLACES"
0160 PRINT "TO THE RIGHT. AFTER THIS STEP X IS EQUAL TO",X*100
0165 PRINT
0170 PRINT "NEXT THE COMPUTER ADDS .5 TO THE COMPUTED VALUE"
0180 PRINT "OF X. AFTER THIS STEP X IS EQUAL TO ",(X*100+.5)
0190 PRINT
0200 PRINT "NEXT THE COMPUTER WILL COMPUTE THE INTEGER OF X"
0210 PRINT "THE INTEGER VALUE OF X IN OUR EXAMPLE IS",INT(X*100+.5)
0220 PRINT
0230 PRINT "IN THE FINAL STEP THE COMPUTER WILL DIVIDE X BY"
0240 PRINT "100 TO MOVE THE DECIMAL PLACE BACK TO WHERE IT"
0250 PRINT "BELONGS. THE FINAL COMPUTED VALUE OF X IS      ";
0251 PRINT INT(X*100+.5)/100
0260 IF Y >1 GOTO 600
0265 PRINT
0300 PRINT "IN THIS RUN THE THOUSANDTH DIGIT OF THE"
0310 PRINT "NUMBER WE ROUNDED WAS A 5 ANY DIGIT 5"
0320 PRINT "OR GREATER IN THE THOUSANDTHS PLACE WILL"
0330 PRINT "CAUSE A CARRY OVER TO THE NEXT MOST"
0340 PRINT "SIGNIFICANT PLACE WHEN WE ADD THE .5"
0350 PRINT
0360 PRINT "NOW LET'S TRY ANOTHER ONE. THIS TIME WE"
0370 PRINT "USE A THOUSANDTH DIGIT LESS THAN 5."
0380 IF Y <2 GOTO 105
0600 PRINT "THIS TIME THE THOUSANDTHS DIGIT WAS LESS"
0610 PRINT "THAN 5 SO WHEN WE ADDED .5 THERE WAS NO"
0620 PRINT "CARRY OVER TO THE NEXT SIGNIFICANT DIGIT"
0700 PRINT
0710 PRINT
0715 PRINT "AND THERE YOU HAVE IT. TRY IT YOURSELF"
0720 PRINT "IN A PROGRAM. GOOD LUCK . . . "
0999 END

READY
#
```

Fig. 1.

Program

... simplifying the decimals

Some of the shorter versions may not allow you to make use of the formula as they do not include the precision arithmetic functions. For example, I have Tiny BASIC which only works with integers.

Fig. 2 is an actual run of the program of Fig. 1. As can be seen, the run starts out by stating that the program is a number rounding routine. It next tells that it is going to provide a helpful formula. About halfway down in Fig. 2 the computer asks for a number to be rounded out. The number is entered and the program describes each step in its operation. This is followed by an explanation to further clarify what was done. Next, another number is requested and input. This second number is then rounded out and explained just as the first one was.

One thing that the run of the program may not make too clear to nonmathematicians such as myself, is the key to the formula. The key is to request the integer (there I go speaking mathematics) of the number to be rounded out. Multiply your number or variable by 100 then add 0.5. Finally divide by 100. Don't forget to use the parenthesis.

I realize that in the examples used in the run of the program, I could have simply added 0.005 and would have come up with the same end result. I have tried this in several programs and have found that it is not always reliable. I prefer to use this longer manipulation as I have had very good luck with it.

So, with that I will let the run of the program finish the explanation. ■

```
RUN

              NUMBER ROUNDING ROUTINE
                WRITTEN IN BASIC

HERE IS A FORMULA WHICH IS VALUABLE IF YOU WANT TO
ROUND OUT A NUMBER IN A PROGRAM. THE FORMULA
IS AS FOLLOWS;

        LET X=INT(N*100+.5)/100

EVEN MORE COMPLEX FORMULAS CAN BE ROUNDED SUCH AS;

        LET X=INT(N*N/N*100+.5)/100

THE VARIABLE N IN THE ABOVE CAN BE ANY NUMBER OR
VARIABLE YOU CHOOSE TO USE.

TO SEE HOW IT WORKS THE COMPUTER WILL ASK FOR A NUMBER
TO BE ROUNDED. THEN IT WILL ROUND THE NUMBER EXPLAINING
EACH STEP AS IT GOES THROUGH IT. HERE WE GO.

WE WILL USE X AS OUR VARIABLE IN THE RUN

INPUT THE NUMBER YOU WANT ROUNDED OUT          ? 21.3456789

FIRST THE COMPUTER MULTIPLIES X BY 100
THIS SHIFTS THE NUMBER TWO DECIMAL PLACES
TO THE RIGHT. AFTER THIS STEP X IS EQUAL TO          2134.56789

NEXT THE COMPUTER ADDS .5 TO THE COMPUTED VALUE
OF X. AFTER THIS STEP X IS EQUAL TO                  2135.06789

NEXT THE COMPUTER WILL COMPUTE THE INTEGER OF X
THE INTEGER VALUE OF X IN OUR EXAMPLE IS             2135

IN THE FINAL STEP THE COMPUTER WILL DIVIDE X BY
100 TO MOVE THE DECIMAL PLACE BACK TO WHERE IT
BELONGS. THE FINAL COMPUTED VALUE OF X IS            21.35

IN THIS RUN THE THOUSANDTH DIGIT OF THE
NUMBER WE ROUNDED WAS A 5. ANY DIGIT 5
OR GREATER IN THE THOUSANDTHS PLACE WILL
CAUSE A CARRY OVER TO THE NEXT MOST
SIGNIFICANT PLACE WHEN WE ADD THE .5

NOW LET'S TRY ANOTHER ONE. THIS TIME WE
USE A THOUSANDTH DIGIT LESS THAN 5.

INPUT THE NUMBER YOU WANT ROUNDED OUT          ? 21.3445678

FIRST THE COMPUTER MULTIPLIES X BY 100
THIS SHIFTS THE NUMBER TWO DECIMAL PLACES
TO THE RIGHT. AFTER THIS STEP X IS EQUAL TO          2134.95678

NEXT THE COMPUTER WILL COMPUTE THE INTEGER OF X
THE INTEGER VALUE OF X IN OUR EXAMPLE IS             2134

IN THE FINAL STEP THE COMPUTER WILL DIVIDE X BY
100 TO MOVE THE DECIMAL PLACE BACK TO WHERE IT
BELONGS. THE FINAL COMPUTED VALUE OF X IS            21.34

THIS TIME THE THOUSANDTHS DIGIT WAS LESS
THAN 5 SO WHEN WE ADDED .5 THERE WAS NO
CARRY OVER TO THE NEXT SIGNIFICANT DIGIT

AND THERE YOU HAVE IT. TRY IT YOURSELF
IN A PROGRAM. GOOD LUCK . . .

READY
#
```

Fig. 2.

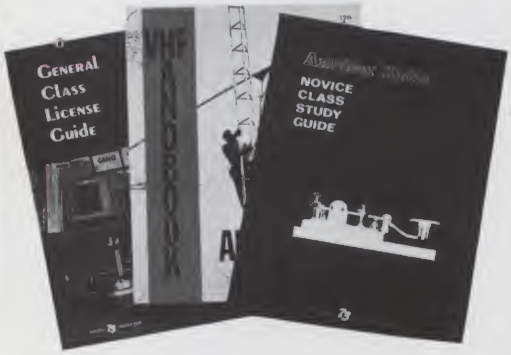
KB BOOK NOOK

Test Equipment Library

● **VOL. I COMPONENT TESTERS** Build your own test equipment and save a bundle (and have a lot of fun). Volume I of the 73 Test Equipment Library shows you how to build and use transistor testers (8 of 'em), three diodes testers, 3 IC testers, 9 voltmeters and VTVMs, 8 ohmmeter, 3 inductance meters, and a raft of other gadgets for checking temperature, crystals, Q, etc. $4.95

● **VOL. II AUDIO FREQUENCY TESTERS** If you're into audio . . . such as digital cassette recording, RTTY, Baudot vs ASCII, SSTV, SSB, Touchtone or even hi-fi . . . you'll want to have this book full of home built test equipment projects. Volume II $4.95

● **VOL. III RADIO FREQUENCY TESTERS** This is of more interest to hams and CBers . . . test equipment you can build for checking out transmitters and receivers: signal generators, noise generators, crystal calibrators, GDOs, dummy loads . . . things like that. This is Volume III of the 73 Test Equipment Library $4.95



● **NOVICE STUDY GUIDE** This is the most complete Novice study guide available. It is brand new. This is not only invaluable for anyone wanting to get started in amateur radio, but also it is about the only really simple book on the fundamentals of electricity and electronics. And without your fundamentals down pat, how can you go on to really understand and work with computers? First things first. $4.95

● **GENERAL CLASS STUDY GUIDE** This book takes over on theory where the Novice book leaves off. You'll need to know the electronic theory in this to work with computers and you'll not find an easier place to get the information. It will also make getting your Tech or General license a breeze . . . then you can get on the ham repeaters and interconnect your micro with others. $5.95

● **VHF ANTENNA HANDBOOK** The NEW VHF Antenna Handbook details the theory, design and construction of hundreds of different VHF and UHF antennas . . . a practical book written for the average amateur who takes joy in building, not full of complex formulas for the design engineer. Packed with fabulous antenna projects you can build. $4.95

● **WEATHER SATELLITE HANDBOOK** Simple equipment and methods for getting good pictures from the weather satellite. Antennas, receivers, monitors, facsimile you can build, tracking, automatic control (you don't even have to be home). Dr. Taggart WB8DQT $4.95

● **SSTV HANDBOOK** This excellent book tells all about it, from its history and basics to the present state of the art techniques. Contains chapters on circuits, monitors, cameras, color SSTV, test equipment and much more. Hardbound $7, Softbound $5

● **RF AND DIGITAL TEST EQUIPMENT YOU CAN BUILD** RF burst, function, square wave generators, variable length pulse generators — 100 kHz marker, i-f and rf sweep generators, audio osc, af/rf signal injector, 146 MHz synthesizer, digital readouts for counters, several counters, prescaler, microwavemeter, etc. 252 pages. $5.95

● **WHAT TO DO AFTER YOU HIT RETURN** PCC's first book of computer games . . . 48 different computer games you can play in BASIC . . . programs, descriptions, muchly illustrated. Lunar landing, Hammurabi, King, Civil 2, Qubic 5, Taxman, Star Trek, Crash, Market, etc. $6.95 ppd.

● **101 GAMES IN BASIC** Okay so once you get your computer up and running in BASIC, then what? Then you need some programs in BASIC, that's what. This book has 101 games for you, from very simple to real buggers. You get the games, a description of the games, the listing to put in your computer and a sample run to show you how they work. Fun. Any one game will be worth more than the price of the book for the fun you and your family will have with it. $7.50 postpaid.

● **BASIC** by Bob Albrecht, etc. Self-teaching guide to the computer language you will need to know for use with your microcomputer. 324 pages. This is one of the easiest ways to learn computer programming. $4.95 ppd.

● **TVT COOKBOOK** by Donald Lancaster, describes the use of a standard television receiver as a microprocessor CRT terminal. Explains and describes character generation, cursor control and interface information in typical, easy-to-understand Lancaster style. This book is a required text for both the microcomputer enthusiast and the amateur RTTY operator who desires a quiet alternative to noisy teletype machines. $9.95

● **TTL COOKBOOK** by Donald Lancaster. Explains what TTL is, how it works, and how to use it. Discusses practical applications, such as a digital counter and display system, events counter, electronic stopwatch, digital voltmeter, and a digital tachometer. 336 pages; 5½ x 8½; softbound. $8.95

Use the order card in the back of this magazine or itemize your order on a separate piece of paper and mail to Kilobaud, Peterborough NH 03458. Be sure to include check or detailed credit card information.

Note: Prices subject to change on books not published by 73 Magazine.

KB BOOK NOOK

● **BRAND NEW DICTIONARY** This new microcomputer dictionary fills the urgent need for all computer people, engineers, scientists, industrialists, communications people — as professionals, amateurs, teachers, or students — to become quickly acquainted with the terminology and nomenclature of a new revolution in computer control capabilities in areas that pervade most of man's daily activities.

Over 5000 definitions and explanations of terms and concepts (704 pages) relating to microprocessors, microcomputers and microcontrollers. There are also separate appendices on: programmable calculators; math and statistics definitions; flowchart symbols and techniques; binary number systems and switching theory; symbol charts and tables; summaries of BASIC FORTRAN and APL. In addition there is a comprehensive electronics/computer abbreviations and acronyms section. $15.95.

● **COMPUTER PROGRAMMING HANDBOOK** by Peter Stark. A complete guide to computer programming and data processing. Includes many worked out examples and history of computers. $8.95

● **MY COMPUTER LIKES ME . . . WHEN I SPEAK BASIC** An introduction to BASIC . . . simple enough for your kids. If you want to teach BASIC to anyone quickly, this booklet is the way to go. $2.00 ppd.

● **SCELBI'S GALAXY GAME FOR THE "8008"/"8080"** Here's a new twist in computer games by Scelbi Computer Consulting and Robert Findley. The game, "Galaxy", pits the operator of a spaceship against alien craft, as well as variables such as speed, time and ammunition. No two games are the same! This game is described in *Galaxy Game for the 8008/8080*, published by Scelbi Computer Consulting, Inc. $14.95.

● **SCELBI'S FIRST BOOK OF COMPUTER GAMES** Need a game for your 8008 or 8080 microprocessor? Try *Scelbi's First Book of Computer Games for the 8008/8080* which described three popular games, "Space Capture", "Hexpawn", and "Hangman". Complete flowcharts, logic description, program listing, and instructions are provided. A must for the game freak! $14.95.

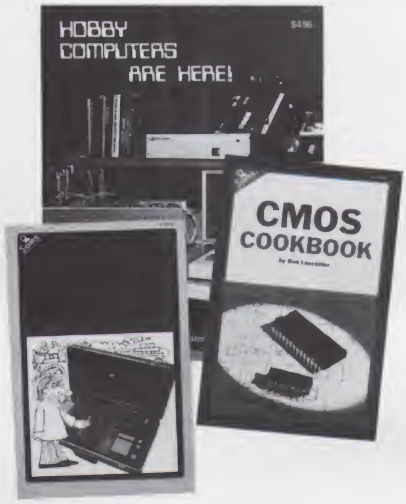
● **6800 SOFTWARE GOURMET GUIDE & COOKBOOK** If you have been spending too much time developing routines for your 6800 microprocessor, try the new book by Scelbi Computing and Robert Findley. This manual, *6800 Software Gourmet Guide and Cookbook* describes sorting, searching, and many other necessary routines for the 6800 user. $9.95.

● **THE STORY OF COMPUTERS** by Donald D. Spencer is to computer books what *Dick and Jane* to novels . . . extremely elementary, gives the non-computerist a fair idea of what the hobbyist is talking about when he speaks computer lingo. Attempts to explain what computers are and can do to a spouse, child or any un-electronics-minded friend. $4.95.

● **CMOS COOKBOOK** by Don Lancaster, pub. Howard W. Sams Company. Another winner from Don Lancaster, author of the famous *RTL* and *TTL Cookbooks*. The *CMOS Cookbook* details the application of CMOS, the low power logic family suitable for most applications presently dominated by TTL. The book follows the style of the original Cookbooks. Eight chapters cover all facets of CMOS logic, and the work is prefaced by 100 pages detailing the characteristics of most CMOS circuits. The *CMOS Cookbook* is required reading for every serious digital experimenter. $9.95

● **MICROCOMPUTER PRIMER** by Mitchell Waite and Michael Pardee, pub. by Howard W. Sams Company. If you are afraid to get involved with microcomputers for fear of not understanding them, worry no longer! The *MICROCOMPUTER PRIMER* describes basic computer theory, explains numbering systems, and introduces the reader to the world of programming. This book does not elaborate on specific systems or chips, but describes the world of microcomputing in "real world" terminology. There is probably no better way of getting involved with the exciting new hobby of microcomputing. $7.95

● **HOBBY COMPUTERS ARE HERE** If you (or a friend) want to come up to speed on how computers work . . . hardware and software . . . this is an excellent book. It starts with the fundamentals and explains the circuits, the basics of programming, along with a couple TVT construction projects, ASCII-Baudot, etc. This book has the highest recommendations as a teaching aid for newcomers. $4.95.



● **THE NEW HOBBY COMPUTERS!** This book takes it from where "Hobby Computers Are Here" leaves off, with chapters on Large Scale Integration, how to choose a microprocessor chip, an introduction to programming, low cost I/O for a computer, computer arithmetic, checking memory boards, a Baudot monitor/editor system, an audible logic probe for finding those tough problems, a ham's computer, a computer QSO machine . . . and much, much more! Everything of interest is there in one volume, ready to be enjoyed by you. $4.95 postpaid.

Use the order card in the back of this magazine or itemize your order on a separate piece of paper and mail to Kilobaud, Peterborough NH 03458. Be sure to include check or detailed credit card information.

Note: Prices subject to change on books not published by 73 Magazine.

Meet the Tarbell/KC Interface

Don Tarbell
Tarbell Electronics
20620 S. Leapwood Ave., Suite P
Carson CA 90746

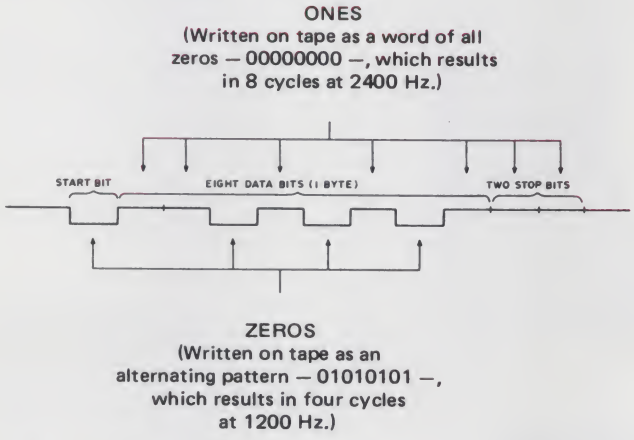


Fig. 1. Kansas City asynchronous format for recording one 8-bit byte.

In November of 1975 there was a meeting in Kansas City, Kansas, of various cassette interface manufacturers to determine a standard for exchange of programs and data on cassettes among computer hobbyists. The format that was proposed as a result

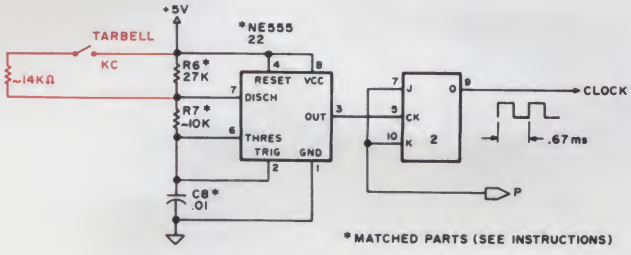


Fig. 2. Output oscillator modification.

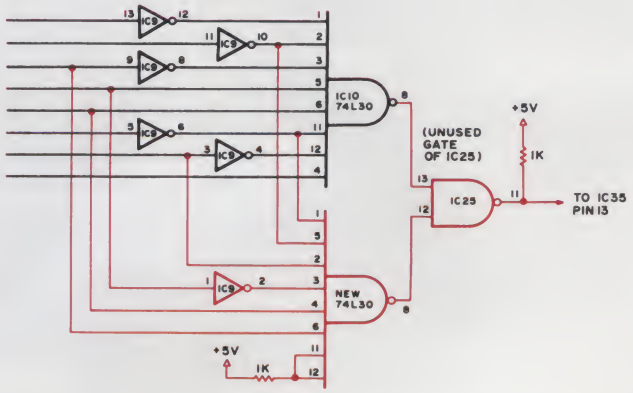


Fig. 3. Input section modification.

of the meeting is a modified version of the coding technique described by Don Lancaster in several publications, and has since become known as the Kansas City Standard. In this format, each 8-bit byte is written on tape in an asynchronous format, with one start bit (zero), 8 data bits (zero or one), and two stop bits (ones). A one is defined as 8 cycles at 2400 bits per second, and a zero is defined as 4 cycles at 1200 bits per second. This provides a data transfer speed of 300 baud, or a little less than 30 bytes per second. Also, the format may be generated and decoded using a variety of techniques.

Since the standard is fairly slow, it suggests that many people may want to have two methods available: one that provides for the Kansas City format, and another that is much faster, to speed program loading and development. Following is a discussion of how the Tarbell Cassette Interface can be modified to provide the 30 bytes per second KC format *and* the faster 187 bytes per second Tarbell format.

The Hardware Modifications

First, the output oscillator frequency will have to be raised from 3000 Hz to 4800 Hz. This is because a higher bit density is required of the tape, although the actual data transfer rate is much slower. A one may be generated by writing a word of all zeros (00000000), and a zero may be generated by writing a word of alternating ones and zeros (01010101). See Fig. 1. An output subroutine converts each byte to be written in this format from parallel to serial form (required only for the KC format).

The frequency of the output oscillator may be changed from its normal value of 3000 Hz to 4800 Hz by putting a resistor of about 14k Ohms in parallel with R6 (27k Ohm).

On the input side, the adjustment of the potentiometer (R8) will have to be changed for the higher frequency. The sync detector circuit (ICs 9 and 10) will have to be changed so that it recognizes the alternating bit pattern as a start bit (or byte) in addition to the normal sync byte of E6 (hex). An input subroutine converts each byte from its serial form to its parallel form (required only for this format).

Using the method outlined above, the Tarbell Cassette Interface can be modified so that a single-pole, single-throw switch will determine which frequency will be used. The software determines the format. Another alternative is to change to the higher frequency permanently, so that no switch is necessary. The disadvantage of *this is that* you would have to readjust the potentiometer to read

```
5F00                    0010 ** KANSAS CITY OUTPUT ROUTINE **
5F00                    0020 ** OUTPUTS THE BYTE IN REGISTER A. **
5F00                    0030 * NOTE:  FREQUENCY OF OUTPUT OSCILLATOR
5F00                    0040 * SHOULD BE CHANGED TO 4800 HZ FOR
5F00                    0050 * OUTPUT FREQUENCIES OF 2400 AND 1200
5F00                    0060 * FOR A 1 AND 0 RESPECTIVELY.
5F00                    0070 *   OCTOBER 27, 1976
5F00 C5                 0080        PUSH    B       SAVE REGISTERS B,C.
5F01 0E 08              0090        MVI     C,8     GET BIT COUNT.
5F03 CD 1B 5F           0100        CALL    ZERO    DO START BIT.
5F06 0F                 0110 LOOP   RRC             LOOK AT LSB.
5F07 DC 15 5F           0120        CC      ONE     IF 1, DO A 1.
5F0A D4 1B 5F           0130        CNC     ZERO    IF 0, DO A 0.
5F0D 0D                 0140        DCR     C       DECREMENT COUNTER.
5F0E C2 06 5F           0150        JNZ     LOOP    DO ALL 8 BITS.
5F11 CD 15 5F           0160        CALL    ONE     DO TWO STOP BITS.
5F14 C1                 0165        POP     B       RESTORE B,C.
5F15 F5                 0170 ONE    PUSH    PSW     SAVE REGISTER A.
5F16 3E 00              0180        MVI     A,0     GET 00000000 PATTERN.
5F18 C3 1E 5F           0190        JMP     REST    DO REST OF IT.
5F1B F5                 0200 ZERO   PUSH    PSW     SAVE REGISTER A.
5F1C 3E 55              0210        MVI     A,55H   GET 01010101 PATTERN.
5F1E CD 23 5F           0220 REST   CALL COUT        OUTPUT TO CASSETTE.
5F21 F1                 0230        POP     PSW     RESTORE REGISTER A.
5F22 C9                 0240        RET             RETURN.
5F23 F5                 0250 COUT   PUSH    PSW     SAVE REGISTER A.
5F24 DB 6E              0260 OLOOP  IN      CASC    READ CASS. STATUS.
5F26 E6 20              0270        ANI     20H     LOOK AT OUT RDY BIT.
5F28 C2 24 5F           0280        JNZ     OLOOP   WAIT TILL READY.
5F2B F1                 0290        POP     PSW     RESTORE REGISTER A.
5F2C D3 6F              0300        OUT     CASD    OUTPUT TO CASSETTE.
5F2E C9                 0310        RET             RETURN FROM COUT.
5F2F                    0320 CASC   EQU     6EH     CASS STATUS PORT.
5F2F                    0330 CASD   EQU     6FH     CASS DATA PORT.
5F2F                    0340 PSW    EQU     6
5F2F                    0350 SP     EQU     6
?
```

Program A. KC format output routine.

```
5F40                    0010 ** KANSAS CITY INPUT ROUTINE **
5F40                    0020 ** READS ONE BYTE INTO REGISTER A. **
5F40                    0022 * NOTE: IN ORDER TO USE THIS ROUTINE,
5F40                    0025 * MAKE THE NECESSARY CHANGES ON YOUR BOARD.
5F40                    0026 *
5F40                    0030 *
5F40 C5                 0034        PUSH    B       SAVE B,C.
5F41 0E 08              0035 BLIN   MVI     C,8     SET COUNT=8 BITS.
5F43 3E 10              0038        MVI     A,10H   RESET RECEIVER.
5F45 D3 6E              0039        OUT     CASC
5F47 CD 73 5F           0040 BLOOP  CALL    CASIN   READ A BYTE (BIT).
5F4A E6 3C              0050        ANI     3CH     LOOK AT MIDDLE 4 BITS.
5F4C CA 65 5F           0060        JZ      ONE     IF XX0000XX, BIT=1.
5F4F FE 3C              0070        CPI     3CH     IF XX1111XX, BIT=1.
5F51 CA 65 5F           0080        JZ      ONE
5F54 FE 14              0090        CPI     14H     IF XX0101XX, BIT=0.
5F56 CA 61 5F           0100        JZ      ZERO
5F59 FE 28              0110        CPI     28H     IF XX1010XX, BIT=0.
5F5B CA 61 5F           0120        JZ      ZERO
5F5E C3 41 5F           0130        JMP     BLIN    MUST BE NOISE.
5F61 BF                 0140 ZERO   CMP     A       CLEAR CARRY.
5F62 C3 66 5F           0150        JMP     REST    DO REST LIKE ONE.
5F65 37                 0170 ONE    STC             SET CARRY=1.
5F66 78                 0180 REST   MOV     A,B     GET RESULT.
5F67 1F                 0190        RAR             SHIFT CARRY INTO MSB.
5F68 47                 0200        MOV     B,A     PUT RESULT BACK.
5F69 0D                 0210        DCR     C       DONE WITH BYTE?
5F6A C2 47 5F           0220        JNZ     BLOOP   IF NOT, KEEP READING.
5F6D CD 73 5F           0230        CALL    CASIN   READ AN EXTRA BYTE.
5F70 78                 0240        MOV     A,B     GET RESULT.
5F71 C1                 0250        POP     B       RESTORE B,C.
5F72 C9                 0260        RET             RETURN.
5F73 DB 6E              0270 CASIN  IN      CASC    READ CASS STATUS.
5F75 E6 10              0280        ANI     10H     LOOK AT INPUT BIT.
5F77 C2 73 5F           0290        JNZ     CASIN   WAIT TILL READY.
5F7A DB 6F              0300        IN      CASD    READ DATA BYTE.
5F7C C9                 0310        RET     RETURN FROM CASIN.
5F7D                    0400 CASC   EQU     6EH     STATUS/CONTROL PORT.
5F7D                    0410 CASD   EQU     6FH     DATA PORT.
?
```

Program B. KC format input routine.

tapes made with the standard 3000 Hz oscillator (187 bytes per second), and that a slightly higher frequency response is required on the part of your recorder.

Fig. 2 is a schematic of the NE555 output oscillator in the Output Section. It illustrates the addition of the switch and the 14k resistor in parallel with R6 to obtain the higher 4800 Hz output. Fig. 3 shows the changes necessary to IC9 and IC10 so the input circuit will recognize the KC format start bit (which will be the alternating bit pattern for a zero). To make these changes, cut the traces going to pins 1 and 2 of IC9, install a 74L30 in one of the spare slots, and make the circuit look like Fig. 3. (This change will not affect the normal operation at 187 bytes/second.)

The Software Modifications

Programs A and B are software routines for output and input of the Kansas City format, respectively. Also, the listings for a Kansas City modified version of ROMP 1 (Read-Only Memory Program) is available from Tarbell Electronics if you will send along an SASE with your request. (ROMP 1 is a program which will allow the user to execute the following functions: begin program execution at a particular address, generate a sync stream, load and execute a program, output a record of a particular length at a particular starting address, input a record of a particular length at a specified starting address, and check the checksum of a cassette record.

Conclusion

If you run into any problems with these modifications, we would like very much to help you out, so don't hesitate to drop a line. We would also enjoy hearing of your successes with running your interface in both formats. ■

Corrections

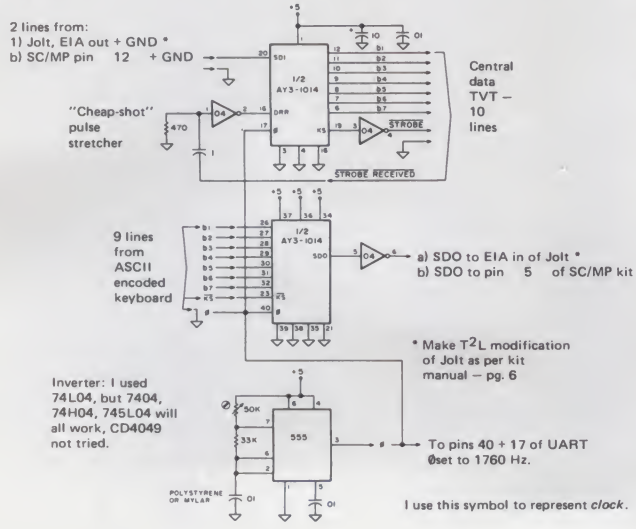
Article by: Bob Grater KILOBAUD ISSUE #1 1595-21 Laurelwood Rd.
A "TTY" Alternative Santa Clara CA 95050

Subject: Interface of "SAB" to Jolt & SC/MP with Jeff Rolaff's central data TVT sub reference: 73 July 1976 – pg. 118.

Problem: SAB did not work.

Analysis: Scope on pin 19 of UART – (I used AY3-1014) revealed a waveform present at all times using circuit as presented in *Kilobaud*.

Solution: Circuit shown below works for interfacing of Jolt & SC/MP to Rolaff's TVT. I was also somewhat hampered by not having a spec sheet for the AY3-1014. Sorry – no circuit board available – the interface was wire-wrapped.



Dear George [Young],

Many thanks for your letter of Jan. 3, you are hereby entitled to one "Gotcha" – I missed the ground on pin 16 of the UART on the redrawn schematic. It's also missing on the kit copies, but not on the PC boards (thank God). Pin 16 should go to ground to enable the DAV line.

Why you had a constant strobe waveform out of pin 19 though is still a mystery to me unless the 74L04 you used was oscillating due to the high values in the pulse stretcher. Incidentally, the change in values required for your version of the stretcher was due to the switch from CMOS to TTL logic.

The change in the 555 R/C network also was a good idea to give a little wider range in Baud Rates.

Don't know if you have a JOLT or not, but you might be interested to know that the DEMON monitor is almost an exact copy of MOS's TIM monitor, and their memory check program works great on JOLT.

Thanks again for your letter and all the trouble you went to drawing it out (makes it much easier to see).

Bob Grater
Santa Clara CA

In "Programming? It's Simple!" (January, page 90), the *Program for finding base, emitter, and collector voltages, emitter current, and gain for the amplifier in Fig.* 5 should be labeled Fig. 6. At LOC 04 delete 02 from Comments. At LOC 05, 20, and 41 add a ÷ to Key and Comments. At LOC 29 Key should be 2 and Comments should be Rc.

In "Super-Tube . . . Jazzing up the Digital Group TVT" (March), the following corrections should be noted: page 124, line 7, column 4 reads Fig. 3 instead of Fig. 2; page 125: Fig. 1, disconnected pins are represented by the dotted line in added socket; Fig. 2, VE-RIFFIC BLANK should read MODIFICATION BLANK; page 126: Fig. 3, the ICP-2 7474 should have a 10 beneath the PRE; column 1, line 10, Return should be capitalized; line 16, should be ICQ-12; the term IC1 on the page should be ICT; the pin numbers should be underscored in column 3, lines 3, 7, 23; column 4, lines 3, 26, 29; also in column 4, line 12, the word is thus instead of this; page 127; column 1, line 12, the word readback should be capitalized; line 47, the pin number should be underscored; page 128: Table 2, column 7, line 48, should read EF instead of EE; Table 3, top portion of table should be moved flush right so that column 7 lines up with the last column of the lower portion; page 129, Table 1, Screen Initialization, pin 13 should be underscored; pin numbers in columns 2 and 3 should all be underscored; page 128, Table 3, blank space between N and n should be a Λ symbol, blank space between L and l should be a \ symbol.

ANNOUNCING OSI 6502 8K BASIC.

OSI's new 8K BASIC for the 6502 was written by Microsoft, the people who wrote ALTAIR® 8K BASIC for the 8080. OSI's 6502 8K BASIC is identical to this powerful and popular 8K BASIC with two very important exceptions: our OSI 6502 8K BASIC has automatic string space handling, and it runs faster. Up to 8 times faster than the 8080 BASIC. And hundreds of times faster than many 6800 BASICs.

In fact, the OSI Challenger with OSI 6502 8K BASIC can actually outperform most small- and medium-scale minicomputers, as well as every micro there is! And that includes the Z-80.

Perhaps even more amazing than its superlative performance is its surprisingly low price: either $50 or free.

OSI 6502 8K BASIC is available to OSI System kit builders for $50 on your choice of paper tape, audio cassette or floppy disk.

And OSI 6502 8K BASIC comes free with the purchase of any 12K or larger OSI Challenger.

So you can own a fully-assembled OSI Challenger complete with serial interface, 12K of RAM memory and our OSI 6502 8K BASIC for just $807.

Incredible!

For more information, contact your local OSI dealer. Write OSI direct for our free OSI brochure. Or enclose $1.00 for the full OSI catalog.

Once again, OSI offers more and costs you less.

OSI

OHIO SCIENTIFIC INSTRUMENTS
Dept. KB
11679 Hayden Street, Hiram, Ohio 44234

IT'S PATRIOTIC TO SUBSCRIBE TO kilobaud MAGAZINE

Driving around trying to locate the latest issue of Kilobaud wastes a lot of gas and helps throw our country on the mercy (or lack of it) of the Arab oil tycoons. Help keep the United States FREE . . . subscribe to Kilobaud.

This will also make sure that you don't miss an issue. On many newsstands, if you don't get there early, you miss out . . . and you are sure to miss the very best issues of KB because they sell out faster than the others.

KILOBAUD IS A BARGAIN AT $15.00 for a one year subscription . . .

You'll save time, energy and money if you subscribe to KB right now! If you have been disappointed in the past by not being able to pick up a copy of KB at your local newsstand, electronics, computer or radio store because they have already been sold out . . . end that emotional trauma now and stop wearing out your car by going store to store in a fruitless hunt for KB once your most convenient outlet is sold out . . . chances are so is everyone else in your neck of the woods. Think of that beautiful $9 savings over the per copy price at the newsstand . . .

KILOBAUD has been out a few months now . . . you and your friends have undoubtedly compared it with other computer hobby magazines . . . you've noticed how jam packed with easy to understand articles it is, you've read Wayne's remarks about the future of hobby computing, you've drooled over the equipment ads, you've noticed how KB appeals to your taste in magazines . . . certainly you don't want to miss a single copy. Subscribe now, and while they last, you can start with the first issue so that your personal library will be complete.

Help others to discover the magazine. If you go around wiping out the newsstand copies every month, you're making it so some other guy will find an empty bin. This could easily disappoint someone to where he might get depressed . . . and commit suicide. You don't want that on your conscience, do you? The safe thing is to subscribe.

This will also save you a bundle. KB is outrageously overpriced on the newsstands at $2 per copy . . . that's $24 for one year. Heck, we'll sell you a one year subscription for only $15, and you'll have $9 to spend for a cruise on a Liberian tanker.

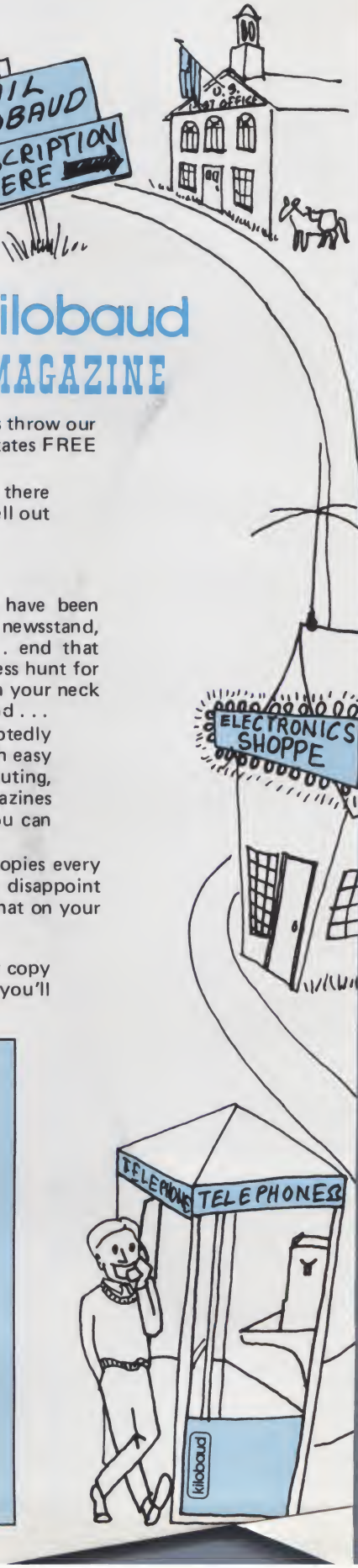
☐ Yes! I want to subscribe to kilobaud
☐ 1 year — $15
Please find $_____enclosed. ☐ Cash ☐ Check ☐ Money Order
Bill: ☐ BankAmericard ☐ Master Charge ☐ American Express
☐ Bill me direct.
*US and Canada only — write for foreign rates

Card # _____ Interbank # _____
Expiration date _____ Signature _____
Name _____
Address _____
City _____ State _____ Zip _____

TOLL FREE SUBSCRIPTION NUMBER (800)258-5473
(allow 6-8 weeks for subscription processing)

K4/77

kilobaud • Peterborough NH 03458



the Computer Store, Inc.

World's Largest MITS/Altair Dealer

COMPUTERS • MITS/Altair 8800B, the world's most popular microcomputer, in an updated, ruggedized configuration.

• MITS/Altair 680-T entry-level BASIC system.

ADD-IN MEMORIES • Available OFF-THE-SHELF for micros and minis!

• RAM & PROM boards for Altair 8800 and 680. Kit or assembled boards . . . 4K dynamics for 680 . . . 4K statics . . . 16K statics . . . 2K PROMS . . .

• 8K and 16K core memories from *Standard Memories* for popular minicomputers: PINCOMM "N", 16K words for NOVA 2 . . . $1,758; PINCOMM "A", 8K/16K words for GA/SPC16 . . . $1,417/1,983; PINCOMM "I", 16K words for Interdata . . . $2,034.

TERMINALS • *Lear Siegler* ADM-3 displays 24 lines of 80 characters on 12" screen . . . $845 kit (participating dealers).

• *Intercolor 8000* displays 25 lines of 80 characters *in color*. Full ASCII plus optional special symbols. Prices from $1,995.

• *DECwriter* II/LA-36DE. Full 128-character ASCII keyboard including upper/lower-case. Prints at 30 cps/132 cols . . . $1,990.

• Teletypes. New & used; ASR/KSR/RO configurations.

PRINTERS • *Centronics* 306-C. Prints at *either* 100 cps/80 cols. *or* 120 cps/96 cols. *or* 165 cps/132 cols. Any *two* combinations in one machine controlled by switch or software command . . . $2,495.

SUPPLIES • Diskettes, cassettes, printer paper and ribbons, you name it. We stock everything we sell, and we sell most everything. Brands include *3M, Information Terminals, IBM.* Write for catalog.

MODEMS • *Universal Data Systems* feature high reliability and built-in diagnostics. Units from $295.

PRODUCTS OF THE MONTH

Special Purchases — Limited Supplies

NEC Single-Board Computers — 8080 systems with serial I/O port, 1K resident monitor, 2K RAM . . . $159.95.

16K Semiconductor Memory for PDP—11/04 & 11/34 . . . $1,595.

Line Printer Controller — PDP11 to LA180 and Centronics printers . . . $695.

Keyboard Terminal — CT256 buffered ASC11 keyboard with 32-character readout and built-in modem. Originally priced at $890, now only . . . $595.

Printer — Beta 30 with RS-232 interface prints at 30 cps/180 cols . . . $795 (used).

The Computer Store accepts both Master Charge and BankAmericard.

____ Charge Master Charge ____ Charge BankAmericard

Sign your name Amount of order

Print name exactly as it is on your card

Good thru Inter Bank number

Enter above the exact number on your card

Your billing address

City State Zip

(Charge customers fill in above)

the Computer Store, Inc.

NAME first name middle initial last name

Rural Route Rural Box No. P.O. Box No.

Company

Street address

City State Zip

Date

The Computer Store, Inc.
[illegible] bridge Street
[illegible] A 01803

The Computer Store
269 Osborne Road
Albany, NY 12211
(518) 459—6140

The Computer Store, Inc.
63 South Main Street
Windsor Locks, CT 06096
(203) 871—1783

The Computer Store of New York (new)
55 West 39th Street
New York, NY 10018
(212) 221—1404

Morris Krieger
37 Eighth Avenue
Brooklyn NY 11217

Super-Tester

... a digital design aid

If you've ever looked through copies of Bugbook I *and* II *by David Larson and Peter Rony (published by E & L Instruments) you were very likely impressed by the digital experiments presented therein. If you've ever looked at the prices of all the hardware (from E & L Instruments) necessary to do those experiments you were very likely impressed by the cost. This is not to say anything against E & L, because I would imagine they were (like most companies) going after the commercial market rather than the less lucrative hobbyist market with their digital design and prototype units. Anyway, it looks like Morris has come up with a solution to the whole problem. As a result of his article, I wouldn't be a bit surprised to see an upsurge in the sales of* Bugbook I *and* II *– John.*

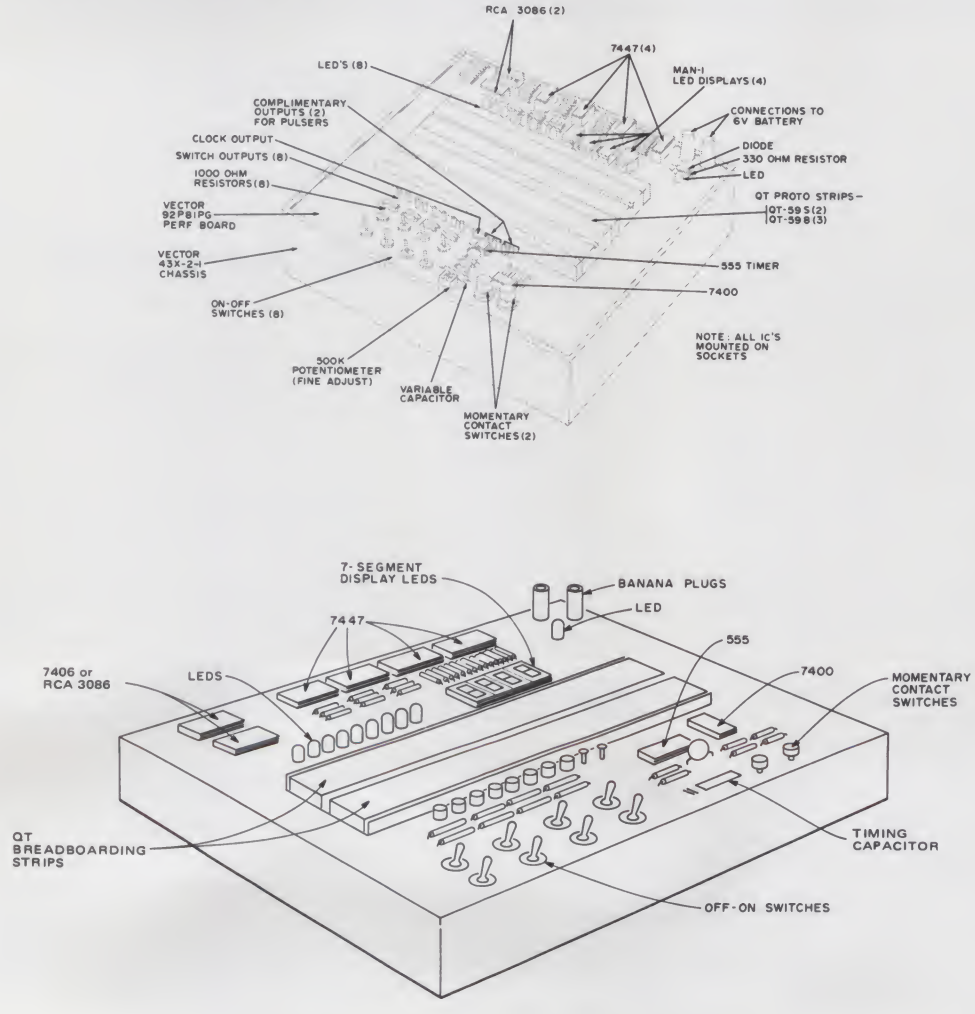


Fig. 1. Suggested layout for breadboard unit.

I am one of the great, and increasing, number of people who are fascinated by microcomputers but who do not know the first thing about digital electronics. When I first heard last summer of the Bugbooks published by E&L Instruments, I jumped to buy them. They seemed to have been written with me in mind — the complete innocent.

I sent away for Bugbooks I, II, IIA, and III, which cost about $30, and discovered that in order to perform the experiments described in Bugbooks I and II, I would have to buy "outboards," little, one-function printed circuits that plugged into a breadboarding socket. Both outboards and socket were sold by E&L Instruments to accompany the Bugbooks.

I sent away for a brochure that described the outboards and found that it would cost me $57.75 for a package of six of these outboards in kit form. The breadboarding socket into which the outboards had to be plugged cost another $19.75. In addition, a package of ICs, jumper wires, resistors, and capacitors, all of which made up "a package of the additional parts required to fully utilize Bugbooks I & II," said the brochure, came to another $40. Thus, the total cost of learning the basics of digital electronics by means of Bugbooks I and II came to $147.50, including the $30 I had spent for the Bugbooks.

At this point I reined in my enthusiasm and decided to have a closer look at the experiments described in the Bugbooks.

I discovered that if I really wanted to "fully utilize" the Bugbooks, I would need not only the six outboards contained in E&L's $57.75 package but the following list of kits and accessories: 2 breadboarding sockets, $39.50 (these prices are all Summer, 1976); 1 power supply module, $6.30; 1 clock outboard module, $9.45; 2 dual pulser modules, $17.90; 2 LED display modules, $23.10; 2 logic switch modules, $14.70; and 4 seven-segment LED display modules, $67.20 — the whole ball of wax coming to $178.15.

Furthermore, the $40 kit of accessories mentioned above includes 24 ICs. A check of all the experiments in Bugbooks I and II showed that I would need not 24 but 57 ICs if I really wanted to "fully utilize" the Bugbooks. When I checked the prices of these ICs in the James Electronics catalog, the total came to $57.69.

All in all, then, to make full use of Bugbooks I and II, I would have had to spend $285.59, a cost that included the cost of the Bugbooks but not the cost of the $40 package of accessory jumpers, resistors, and capacitors.

I concluded this was too high a price to pay in order to learn the basics of digital electronics. I decided instead to try and build an all-in-one breadboarding unit that would include everything required to perform all the Bugbook experiments.

From Sol Libes of the Amateur Computer Group of New Jersey and Christopher Terry of the New York Amateur Computer Club, I obtained schematic diagrams of circuits that were functionally equivalent to those on the outboards. These are the circuits shown in the figures. (By the way, these are not *my* circuits. I still don't know much about digital electronics. I certainly don't know whether these are the best, simplest, or least expensive circuits that will perform the desired functions, but, as far as the experiments are concerned, the circuits have worked.)

Having obtained the circuit diagrams, I put them together into the piece of hardware that is shown in the illustration. The price for all the parts came to about $87.50. Thus, for a total cost of less than one-half the price of the E&L kits, I have a breadboarding system that I think is superior to the group of kits sold by E&L Instruments. I can use it not only to perform the experiments in the Bugbooks but also to set up and experiment with TTL circuits of any kind. At the present time I am limited to TTL circuits because I am using a 6-volt battery as my power supply but, of course, it would be a simple matter for anyone to attach a suitable 115-volt transformer and additional banana plugs to the unit and thus obtain a variety of voltage inputs.

Before I describe the construction, I should say something about the direct cost of my building the breadboarding unit. Since kit-building is new to me, I didn't even own a soldering iron. I have had to buy not only a soldering iron, but also solder, all the essential hand tools, and wire-wrapping* tools as well, and lay in a supply of resistors, capacitors, wire, and much else besides. So, in one sense, my breadboarding system has actually cost me considerably more than the $87.50 I mentioned above. In another sense, however, since I hope to spread the cost of this equipment over a great many circuits and kits that I hope to build over the years, I have not added this expense to the $87.50. However, any beginner who is in my shoes should realize that his start-up costs are likely to be much higher than he anticipates.

*Wire-wrap is a registered trademark of Gardner-Denver Co.

Construction

As may be seen in the diagrams, the circuits are mounted on a large, approximately 8-7/16" x 9½", sheet of P-type vectorboard in which the holes are spaced 0.10" apart. This allowed me to use standard wire-wrap sockets for the ICs and 0.042" Vector wire-wrap posts and pins throughout. I suppose I could have used a soldering technique but it seemed to me to be dumb to go to the trouble of designing, laying-out, and making a printed circuit, especially since I am new at this racket and certain to make any number of damn-fool mistakes. I could have soldered everything together on vectorboard, however, I wanted to be able to undo my mistakes as I discovered them as simply as possible. In fact, I don't think there was one circuit that worked the way it was supposed to the first time I put it together. I made all sorts of simple mistakes, such as confusing IC pin numbers, forgetting to connect the ground wire in a circuit, putting an IC in backwards, and on and on. These gaffes resulted in an unexpected bonus as I learned the rudiments of troubleshooting a digital circuit along with the technique of constructing wire-wrapped circuits. God only knows what sort of job I would have ended up with if I had soldered everything together.

I also learned quickly that one can't troubleshoot a digital circuit using only a volt-ohmmeter, which is the only electrical instrument I had. I had to go out and buy a logic probe (after learning that logic probes existed and being

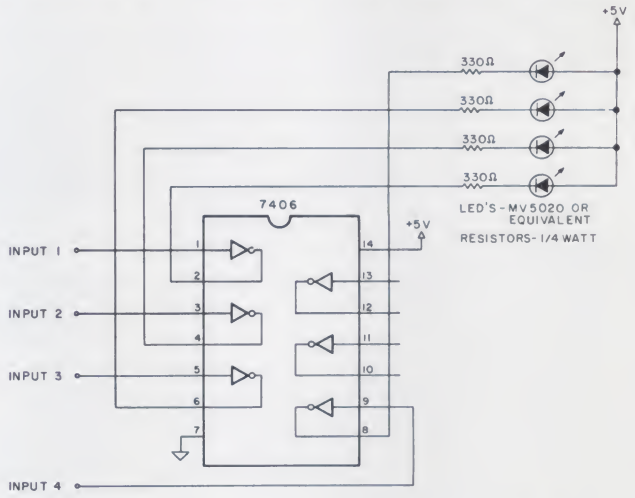


Fig. 2. Lamp driver circuits. These devices prevent the circuit controlling a lamp from being overloaded by the LED. As many of these devices as desired may be incorporated into the breadboard device.

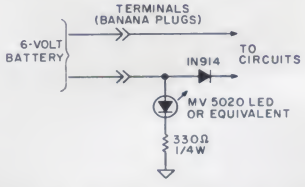


Fig. 3. Power supply for the breadboard tester. The silicon diode drops the voltage from the 6 V battery to about 5.4 V, suitable for TTL circuits. (An ac supply using a LM-309 could also be used.)

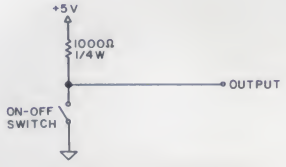


Fig. 4. Simple circuit to provide either a TTL "1" or "0". Any resistor up to 2.7K is suitable for the pull-up.

shown what they could do). Once I appreciated that a digital circuit is in either a voltage or a no voltage condition (how basic can one get?) and that one could analyze any digital circuit in terms of whether or not the voltage in that circuit was high or low (which is also as simple as one can get), troubleshooting my circuits became easy as pie.

The only soldering absolutely required were the connections to the switches. Everything else was wire-wrapped. I discovered that wire-wrapping is simple enough to do if you have the temperament to do the job as it is supposed to be done — which is very carefully. The 30-gage wire used is very fine indeed, it breaks quite easily if it is nicked and then stressed, and the fit between a group of wire-wrap pins is often very close. Wire-wrapping has much more in common with watch-repairing or jewelry-making than any other craft I know of. In fact, I found that the work I was doing was so fine I had to go out and buy a jeweler's loupe in order to check the quality of my work.

Practically, the great virtue of wire-wrapping over soldering, especially to a beginner, is that if one makes a mistake it is simple enough to unwrap the connections and start over again. Once, having completed a circuit, I became dissatisfied with my component layout, unwound all the connections, changed the components into a more satisfying layout, and wire-wrapped everything back together again. You can't do this as easily when you are soldering.

Vector makes a variety of wire-wrap pins and posts for use with their P-type vectorboards, and I used their parts throughout the project. I dare say one can do an adequate job of inserting the pins into the vectorboard merely by pressing them in place with a pair of pliers but I wanted to have a good-looking as well as functional job and so I went to the trouble of using an alignment block (Vector MB45-20-062) and staking tools to make sure the pins were all inserted neatly, securely, and parallel to each other; the alignment block also acted as an anvil as I staked the pins in place.

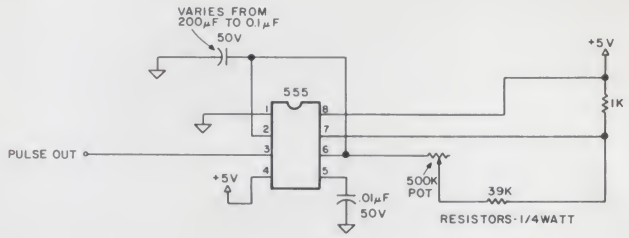


Fig. 7. Clock generator for the breadboard unit. Square wave output appears at pin 3 of the 555 timer, the timing capacitor (200 uF to .1 uF) should be a high quality device.

There are two points about the use of Vector wire-wrap pins the beginner should know about. First, I used type R32 pins as terminals to which I jumpered the wires connecting the circuit components on the breadboard to different switches and LEDs. These are the pins that are located alongside the breadboarding socket in the illustration. The smallest gage wire which can be inserted into these pins is 24 gage, and for this reason you must be sure to have 24-gage wire on hand rather than the usual 22-gage wire.

Second, I used type T49 posts to connect the resistors into their circuits. At first I tried securing the resistor leads by squeezing the split ends of the posts around the leads with a pair of pliers, but I quickly discovered that unless this were done very carefully the post ends acted like jaws of a scissors that neatly clipped the leads in half. I would then have to pull the post out, throw the resistor away, and start over again. Then I tried soldering the resistor leads to the posts. This worked but I didn't want to do any soldering out of principle — besides, what if I wanted to, or had to, change resistor values? I ended up merely pushing the resistor leads into the V made by the post ends. So far, the circuits operate all right even though the resistor leads are in place only finger-tight.

There is little else to say about the construction. I was very careful to check the continuity of all my circuits immediately after I had completed the wire-wrapping, and I was equally careful to check the voltage levels in each circuit after each IC had been inserted into its socket to see if the voltages corresponded to those called for on the circuit diagram. There are no arcane tricks or technical wizardry to wire-wrapping, or even to checking the continuity of a circuit. The more wire-wrapping you do, the better you get at it, though it does require a certain minimal amount of care and attention. Wire-wrapping is not, however, for impatient, nervous types. They should stick to soldering, and they can have it.

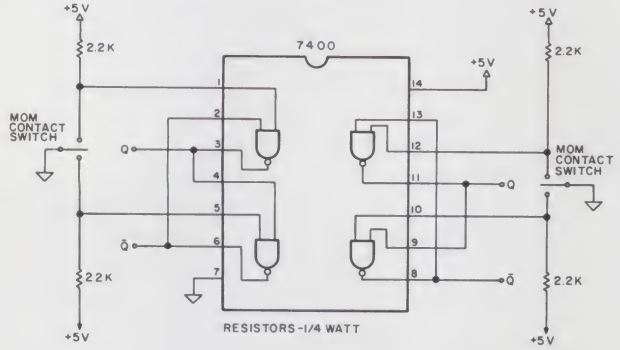


Fig. 5. Dual pulser (bounceless switch). This is an absolutely invaluable part of the unit. Depressing either switch produces a single state change at the Q and $\bar{Q}$ outputs.

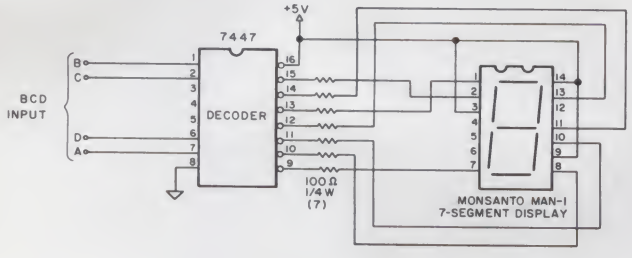


Fig. 6. Seven-segment display and decoder. As many of these modules as desired may be incorporated into the breadboard unit.

Summary

As far as applications are concerned I would, of course, refer you to Bugbooks I and II. If you really want to get down to the nitty-gritty of learning digital electronics I feel those books (along with my unit) will help you achieve the objective.

I'm also using the unit to build circuits from Don Lancaster's TTL Cookbook as well as other circuits which interest me. Naturally, when I'm through with the learning stages I'll be using it to breadboard digital circuits for years to come. ■

ok®

WIRE WRAPPING TOOL
For AWG 30, .025" (0,63mm) sq. post, "MODIFIED" wrap, positive indexing, anti-overwrapping device

OK MACHINE & TOOL CORPORATION
3455 Conner St., Bronx, N.Y.10475 / (212) 994-6600 / Telex 125091

NEW

HOBBY-WRAP
Model BW-630

Battery **wire wrapping** tool

ONLY **$34.95** (batteries not included)

COMPLETE WITH BIT AND SLEEVE

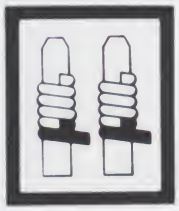
NEW

HOBBY-WRAP
Model BW-630

Battery **wire wrapping** tool

ONLY **$34.95** (batteries not included)

COMPLETE WITH BIT AND SLEEVE



WIRE WRAPPING TOOL
For AWG 30, .025" (0,63mm) sq. post, "MODIFIED" wrap, positive indexing, anti-overwrapping device.

ok®

OK MACHINE & TOOL CORPORATION
3455 Conner St., Bronx, N.Y.10475 / (212) 994-6600 / Telex 125091

Dick Wilcox
1342 Mauna Loa Rd.
Tustin CA 92680

In this third part on developing a do-it-yourself Operating System, Dick continues his easy-going and easy-to-understand delivery of what is normally a very complex subject. Needless to say, once you start developing the system you'll come to appreciate just how complex it really is. Those of you who do take it on as a project please be sure and let us know about your efforts. — John.

The Hobbyist's Operating System

One of the primary functions of any good operating system is to provide the user with a convenient means of communicating with the system and with programs he has written. When no program is running and the system is turned on but sitting idle, it is said to be *waiting for a command* from the user to initiate some specific operation. The task of accepting this command and transforming it into the desired computer function is called *command language processing* and is the subject of this article, the third in a series on operating system practices.

As with almost all problems, there are tradeoffs to be considered when designing any data handling scheme, including the job of having your operating system talk to its users. In this article I will discuss various common methods which have been used in command language processing along with the pros and cons associated with them. I will then go into more detail on a system which is the result of several years of writing and rewriting user-oriented languages. This system has several features which make it handy for small systems and is well within the realm of the hobbyist microcomputer field.

Since this article is only one of a series, there will be some references to processes which will be discussed more fully in future articles. Most of these will be fairly common functions which would normally be found in any advanced operating system such as sending and receiving characters from the user's console terminal and loading program files from disk. Hopefully, in all cases, I will give enough of a functional description of these external routines to enable the big picture to be understood and implemented with a little forethought.

It should be noted that the operating system which we are attempting to develop in this series of articles has a *resident* monitor, which means the user programs do not overlay any portion of the monitor during operation. When the user program is finished it *exits*, which means it returns control to the monitor via some predefined procedure such as jumping to a specific location within the monitor. The monitor then performs some general clean-up functions which may include automatically deleting the user program from memory and returning to *command mode*. This is a mode in which the monitor waits for a command to be typed by the user to initiate the next function. This is what we will be discussing for the next few pages.

Some Common Methods

Traditionally, commands given to the operating system fall into two categories, those that initiate a system function and those that initiate a user function. System functions normally involve general maintenance of the system itself and the management of peripheral data files. Copying, listing, creating, deleting and renaming files are typical of these functions. Operating system generation and modification would also fall into this category. User functions involve the initiation and running of user-written programs and systems such as business applications and games. The editing and assembling of user programs is sometimes up for grabs as to its place in the scheme of things.

The most popular method of handling commands typed in by the user has been to have a small group of predefined code characters or words represent the required system functions. One of these functions then becomes loading a specific user program and optionally initiating its execution from a tape or disk file. One of the easiest schemes to implement is to use a single letter to define the function desired. For example:

D — display memory
E — enter into memory
F — list files on disk
C — create a disk file
T — type out a disk file
X — delete a disk file
L — load a user program
R — run a user program

The above represents a set of typical single letter commands to perform specific system functions. The last two of these represent system functions which result in the initiation of user functions. The pros and cons for this scheme are fairly obvious. The single letter commands are simplest to implement since only one comparison must be made for any new command line entered on the terminal by the user. The commands are usually stored in a table along with the associated addresses for the routines which they invoke. If few commands exist, they might even be coded in the main program flow as a series of direct compares against the single character entered on the terminal. The problem arises that new commands require the direct modification and regeneration of the operating system or at least

... Part 3: Command Language Processing

that portion which processes the user commands. Also, in this scheme the routines that perform the functions are usually a part of the resident operating system which uses valuable memory space. Although this is acceptable and perhaps the most reasonable way for memory based operating systems, we are analyzing techniques for a more advanced system with disk or at least a reasonably fast tape storage media.

Another technique which overcomes the vagueness of the single letter commands is the use of code words which are usually restricted to a maximum size, typically 4 letters. This makes the commands easier to remember but more difficult to implement in the monitor. The same commands in the preceding example might appear as:

DISP — display memory
ENTR — enter into memory
FILE — list files on disk
CREA — create a disk file
TYPE — type out a disk file
DEL — delete a disk file
LOAD — load a user program
RUN — run a user program

The first apparent advantage to this over the single letter commands is that the two commands DISPLAY and DELETE can now be made unique even though they start with the same letter. We do not have to substitute the letter X for the delete command which may have caused some confusion. These multiletter commands do require more extensive programming efforts to decode. The single letter comparisons must be replaced by a more extensive string comparison which involves several steps on most computers. Also, a table lookup scheme is definitely required for matching, and of course, the commands in the table take up more memory than single characters.

Other variations on the above schemes may include longer more meaningful command words, several series of command words resembling English sentences, and command modifiers which allow the same command to perform multiple functions when invoked. These schemes are normally employed in large scale computer systems for specific applications or environments and pose no significant advantages to the hobbyist in a smaller system. No further mention will be made here concerning them (time and space are money).

A More Flexible Approach

The major concern over the aforementioned command language processors is the amount of effort required to implement a new command into your operating system. Even if you wrote the system yourself, it is a bother to modify it, and if it is someone else's pride-and-joy, it may be all but impossible to add a new command. Since the hobbyist has traditionally been in a continual mode of improvement, this is a serious drawback. An ideal situation would be a means of implementing new commands without including them in the operating system itself and without the necessity of reassembling or even regenerating the system proper. Various tricky methods have crossed my path and sometimes have worked their way into one or more of my older system designs. These methods included command tables which were external to the operating system itself along with a multitude of intricate programs to adjust and maintain a linking scheme to insure they eventually wound up performing the desired function. I shudder every time I drag out an old system flowchart just for nostalgia's sake. Some of them would have made all but the most avid programming buff shake his head and sigh. (Been there yourself once or twice, have you?)

Suppose, instead of incorporating a group of absolute commands into the operating system and also including the routines themselves, we accept one full line of input from the user terminal and then treat the first word in that line as the command to be performed. That word would then be interpreted as a program name which must exist in some available form and that program in turn would be automatically loaded and started by the operating system. This now means that system functions as well as user functions will exist as separate programs which will be invoked in exactly the same manner. The main advantage here is the ease of implementing a new command in the system which merely involves writing the program to perform the function, assembling it into runnable format, and loading

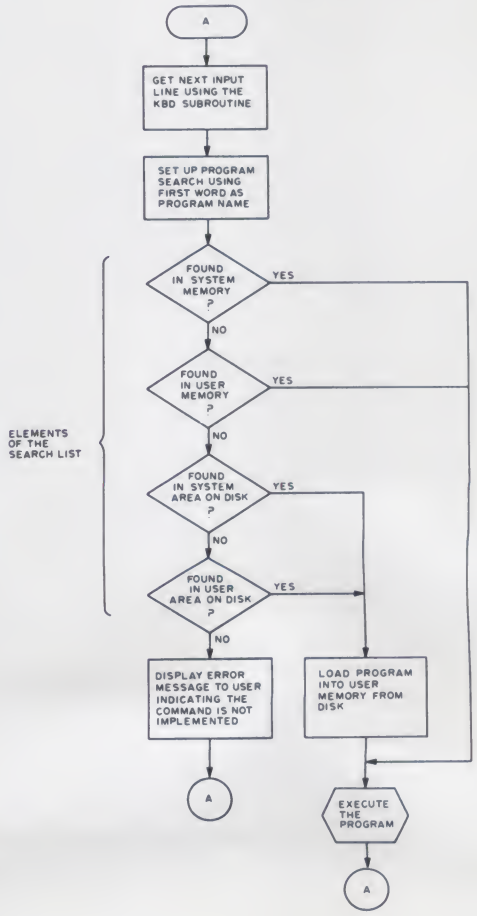


Fig. 1. Basic command processing flow.

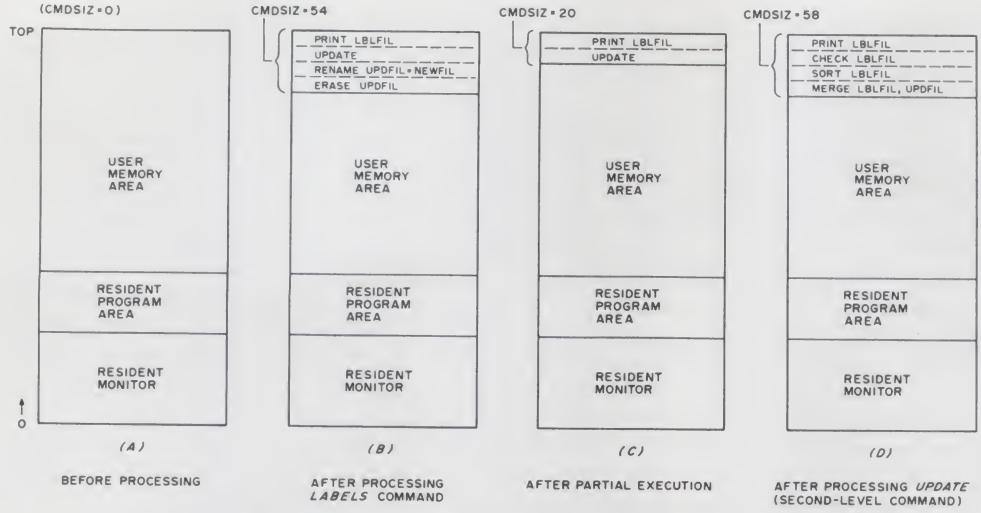


Fig. 2. Memory layouts during command file processing.

it onto the system disk or utility load tape as required.

A secondary advantage is the minimal amount of actual coding in the resident operating system itself required to process the command. The major routines necessary are the terminal I/O routine for accepting the command, a reformatter if the command is not in correct format for your particular I/O scheme, and the program loader to load from disk or tape. If your system supports multiple programs in memory at once, you may even preload all expected programs and then search for them as the commands are entered.

The system that will be developed in the remainder of the article is based on the above technique which I have been using in one form or another for the past five years or so in various systems. It is organized around a disk storage device from which programs may be automatically loaded but the techniques employed will work on tape systems with some limitations. It contains some features which may be used or omitted at your discretion without seriously impairing overall design goals.

Locating The Command Program

Once the command has been entered by the user (terminated by a carriage-return) the operating system must locate the program whose name matches the first word in the input line. Where this program exists and how it is loaded and then executed will depend a lot on the basic organization of your memory and I/O system. I am going to present some ideas which may or may not be totally suited to your monitor but the basics are valid.

There are some programs which under certain conditions should be in memory on a permanent basis and not be *transient*. The term transient is used in operating system nomenclature to refer to a routine or program which is normally stored on disk, loaded into main memory only when needed, and automatically deleted from memory when execution is done. Although most of the command programs are of a transient nature, some programs do not lend themselves to this scheme. Take, for instance, a program which continually updates a dynamic video display during the execution of other tasks (perhaps on an interrupt basis). Execution of this program might only set up the parameters for the display and the refreshing then proceeds automatically. We cannot let the program be deleted or the refreshing would be aborted and possibly cause system failure. In this case the program might be stored in a special area of the resident monitor where it is executed freely without being deleted after it exits. We will call this special area the *resident program area*. Programs must be set up in this area when the monitor is generated or a dynamic scheme must be devised to add them by the user after the monitor is started. In my current system, I incorporate these programs when I generate the monitor. This produces the fewest headaches.

A good operating system should allow the user to load one or possibly more than one program into his work space in main memory where that program may be modified for debugging purposes before being executed. Some system command such as LOAD or GET could be used to accomplish this and the program must then be locatable again by its name. The ability to load up memory with multiple programs ready for execution usually involves assembling each specific program to reside in a different area or using a microprocessor which supports totally relocatable code.

Relocatable, as used here, means the program code will operate properly anywhere in memory without modification or reassembly. Generally speaking, the 8080, Z-80, and 6800 series do not support totally relocatable code. Programs which are loaded in user memory and referenced by their names are considered *resident* as opposed to *transient* since they will not be automatically deleted when they exist.

Programs which are not already in memory (either monitor or user area) must be loaded in from disk for execution. As with memory, a good operating system will normally have at least two distinct areas on disk for data and program storage. One area will be reserved for the operating system itself and the system programs. The other area will be used for work storage and program development. More advanced systems may even have multiple distinct user areas on disk for the separation of users by account number. The system we will describe here will assume one system library area (for system programs) and one user area (for user development programs). The location of a program not already in memory would then proceed logically to one of the areas on the disk.

A good operating system should allow the user to load one or possibly more than one program into his work space in main memory

We have now outlined several areas where the desired program (named in the command line) may be found for execution. The next question that arises is how do we logically search for this program. Obviously we cannot look in all places at once and what do we do if the program

is found in more than one area? The answer to these questions lies in a technique which we shall refer to as a *search list*. This search list specifies the order in which the command processor will search for the requested program. Even though this search scheme could merely be put into direct code, the incorporation of an ordered list which directs the search lends flexibility to the overall system. For instance, the list could be made available for modification by user commands to change the search order dynamically. Also, if you get into multiuser or timesharing applications, each user could have his own personal search list for processing directive. The actual implementation of the search list is dependent upon overall operating system structure and will be left to the ingenuity of the reader.

Fig. 1 depicts the flowchart of our command file processor thus far and shows the reference to four areas within the search list. Entry at point A is made each time the system returns to command mode to get the next command for processing. Once the command has been accepted the program search is made in the order specified by the search list. If the program is located in one of the memory areas it is assumed to be ready for execution and is merely started. If the program is located on disk it must be first loaded into memory before it can be started. We must assume that if the program is not located in any of the specified areas, the command is illegal or not implemented in the current system. In this case the command processor should display some brief error message to the user and return to command mode to await the next command.

Delivering Parameters To The Program

We have seen how the first word on a command line can be used to locate a program with the same name for processing. Many of these programs, however, require one or more parameters to direct their execution. A program which erases files from the disk might be called by the ERASE command but the program will still need to know the name of the files to be erased. One method commonly used is to have the ERASE program ask for the file names after it has been initiated. Another method that I have been using successfully can be more meaningful in our command system. Before the command language processor begins analyzing the command to search for the program, one full input line will be accepted from the user and built up in the terminal input buffer. Even though the command processor only uses the first word to locate the program, the user may type the required parameters (in this case the file names to be erased) on the same line following the command word. If the command processor sets a specific index register to the first word following the actual command word, the program may then use this index to pick up the parameters out of the input line buffer and proceed with its execution. Hence, we then have the command:

```
ERASE FILE1, FILE2, TEST B
```

This command would erase the three files named FILE1, FILE2, and TESTB. The command itself becomes almost self-explanatory and easy to remember. Other commands which work in a similar fashion might be:

```
ASSEM FILE1
LIST FILE1
DISPLA 3446
```

Using our system as we have developed it so far, we see how easy it might be to implement a command to print menus for specific days of the week. Assuming you have a program which needs the name of the day to load up and print a menu for, you might call the program MENU and have it use the command index register to pick up the name of the day from the command input line. The command to print the menu for Monday would then become:

```
MENU MONDAY
```

This command is fairly straightforward even for the most inexperienced computer user. Implementation of the command was simple and required no modifications to the operating system itself. Development of the program and putting it onto the disk (in either the system or the user area) was all that was necessary to define the MENU command and incorporate it into the system. Conversely, removing a command is as simple as erasing the program from the disk.

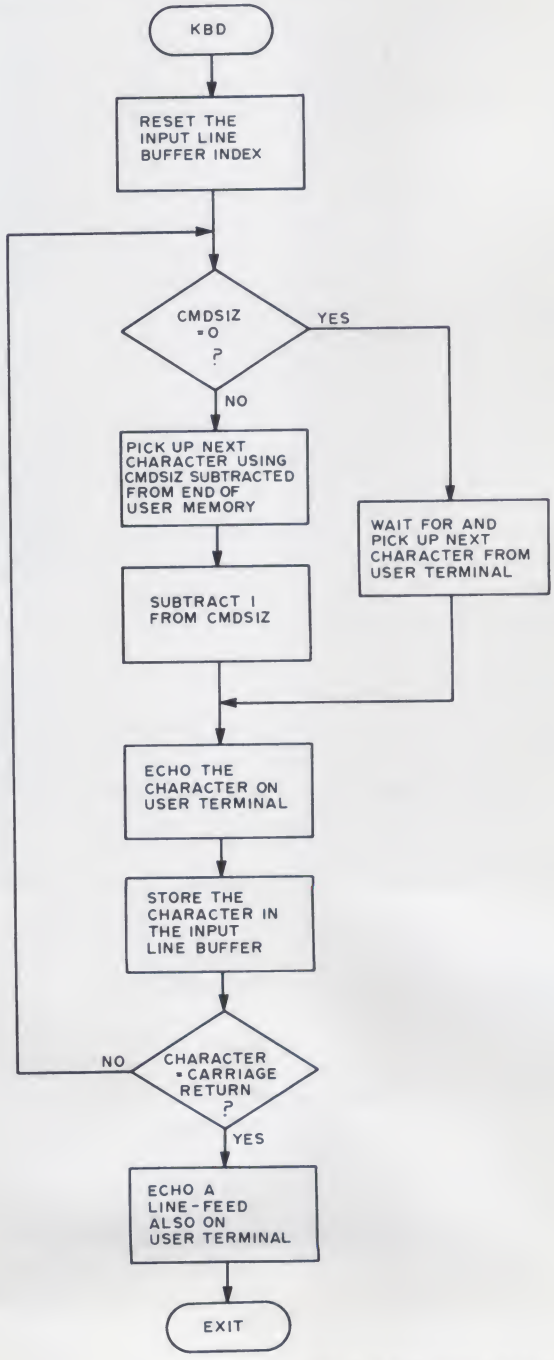


Fig. 3. KBD input line subroutine.

Introduction Of The Command File Concept

In the preceding sections I have described a very useful and easily implemented scheme for processing single line commands and invoking the program which performs the function. Although this system is complete in itself and will accomplish the objective of analyzing and initiating system commands, we can increase its flexibility and usefulness tremendously by the addition of a *command file* processor.

A command file is defined as an ASCII text file on disk (or in memory) composed of one or more lines, each line being a *command line*. A command line is any line which is a legal system or user command that would produce a valid function if it were typed into the system on the user terminal. As each command line is encountered during the processing of the command file, that command sequence is executed by the system just as if it had been typed in from the user's terminal. Command lines may be normal program name commands or they may be other command file names in which case they are said to be *nested*. Nesting occurs when one command file calls other command files within its own normal sequence of processing.

Command files add to the system versatility by allowing a command to be quickly implemented which causes one or more programs to be executed in a specified sequence.

Command files add to the system versatility by allowing a command to be quickly implemented which causes one or more programs to be executed in a specified sequence. In order to implement command file processing in our system we must alter the action taken when the file we are searching for is found (Fig. 1). There must exist some method to detect that the file located is an ASCII command file and not a runnable program file. This may be a code character within the name itself or possibly a flag in the file or file directory block. In any case, instead of executing the program once it has been found, the command processor must move the ASCII lines in the command file up to the current end of the allocated user memory. We will assume that the operating system knows exactly where this is.

In describing the processing of command files, several actions all depend on one another for proper results. I will attempt to explain this in a logical manner and hope that you will bear with me through this, remembering that I am not a college professor, but a hobbyist like yourself. In laying out the sequence of events I will be using a hypothetical situation using two nested command files which in turn command the execution of several programs. I will be referring to Fig. 2 which gives the memory layouts for four key points along the way and also to Fig. 3 which is the flowchart for the basic KBD line input subroutine. Table 1 describes the names of files, programs, and routines which will be referred to also. Remember that the two command files and the programs called and executed are merely an example and bear no other importance to the system we are developing. The real units are the KBD subroutine and the CMDSIZ counter, both of which are a part of our command language processor.

LABELS — a command file consisting of the ASCII command lines:

```
ERASE UPDFIL
RENAME UPDFIL=NEWFIL
UPDATE
PRINT LBLFIL
```

UPDATE — a command file consisting of the ASCII command lines:

```
MERGE LBLFIL, UPDFIL
SORT LBLFIL
CHECK LBLFIL
```

ERASE — system program which erases files from disk.
RENAME — system program which renames a file on disk.
PRINT — system program which prints a disk file on the printer.
MERGE — user program which merges two disk files.
SORT — user program which sorts a disk file.
CHECK — user program which checks a sorted disk file for duplicates.
KBD — subroutine which gets one input line from user terminal or command file.
CMDSIZ — work counter which keeps track of current command area size.

Table 1. Nested command files for processing example.

When the user enters the LABELS command, our command processor locates the LABELS command file on disk and loads it into memory. However, instead of executing it as a program, the processor moves it up to the top of user memory. Refer to Fig. 2. Segment A shows the memory layout before any commands are entered and segment B shows the layout after we have loaded the LABELS command file up to the top. The CMDSIZ work counter initially starts at zero (no command file in progress) and we now add in the total number of bytes in the command file. Actually, when moving the command file data up to the top of memory, we really moved it up to the base of any currently stored command represented by the current value of CMDSIZ. Since this is the first level command file, CMDSIZ is initially zero and LABELS does indeed get moved to the top of user memory. By knowing the current value of CMDSIZ and subtracting it from the end-of-memory address, we can locate the current starting address of the stored command lines waiting to be processed. In our example we have just stored 4 lines of data for a total of 54 bytes including the 4 carriage-returns. CMDSIZ therefore now contains the value 54.

Once the command file has been moved to upper memory it is deleted from its original spot as loaded from disk, and control is returned to the command mode routine (point A of Fig. 1). Previously, it was assumed that each time we entered command mode the KBD subroutine would wait for and accept one full line of input data from the user terminal. This line would be the next command to the system. However, since CMDSIZ is no longer zero, we have at least one command line waiting for processing stored in upper memory. Refer to Fig. 3. The KBD subroutine will now get the input line from upper memory instead of from the user terminal. CMDSIZ will be decremented as each character is moved to the input line, thereby effectively deleting the line from its position in upper memory. When KBD exits back to the command processor (Fig. 1) it appears just as if the line had been entered on the user terminal and processing continues. This action will repeat itself until the last command line has been processed. When this happens, CMDSIZ will have been decremented to zero and the next call to KBD will again cause the input to be requested from the user terminal.

To show the processing of

a nested command file we refer back to Fig. 2 again. Segment C shows the memory layout after the first two command lines have been processed (the ERASE and RENAME commands). CMDSIZ has been decremented to a count of 20 which represents the remaining two commands (UPDATE and PRINT). The UPDATE command, however, is another command file instead of an executable program. Instead of executing it once we locate it, we must move it to upper memory under the rules stated previously. In this case we still have one line left to be executed and CMDSIZ contains the value of 13. (Remember the KBD subroutine deleted the UPDATE command line as it moved it to the input line buffer). Since we will load the new UPDATE command file data (3 lines worth) at the address obtained by subtracting CMDSIZ from end-of-memory, the UPDATE file will be added to the existing command line (PRINT). The resulting memory map is depicted in segment D of Fig. 2. Processing then continues as previously described for the remaining 4 command lines (MERGE, SORT, CHECK, PRINT) which are all programs and not command files.

To recap the sequence of events that will occur in our example:

1. The user enters the command LABELS on the terminal.
2. The LABELS command file is loaded and moved to upper memory.
3. The ERASE program erases the old UPDFIL file.
4. The RENAME program renames the NEWFIL file to UPDFIL.
5. The UPDATE command file is loaded and moved to upper memory.
6. The MERGE program merges the UPDFIL file into the LBLFIL file.
7. The SORT program sorts the merged LBLFIL file.
8. The CHECK program checks the sorted file for duplicates.
9. The PRINT program prints the resulting label file on the printer.
10. The monitor returns to command mode waiting for user terminal input.

As you can see, this method of laying out memory lends itself very nicely to commands which may be nested to any level (limited only by the amount of user memory available). No fancy linking schemes are required to link one command file to the next and only the currently active commands waiting to be processed are actually occupying user memory space. The method I used to scurry you through the operations performed may require pondering over a few times but remember that this system has evolved over a period of years from trial and error. To describe it in detail for the various machines that have already infiltrated the hobbyist market would take several articles in itself. I am hoping that enough detail has been given here to present the overall picture, allowing you to fill in the holes and tailor it to your own needs and hardware. Please recall that as mentioned in my introductory article (*Kilobaud #1*), this series is designed to spur your interest in designing and customizing your own personal operating system and not to impose some rigid rules or absolute program coding on you.

We have developed here the basis for a system which allows new commands to be added to the system with no modification to the monitor.

Bells And Whistles

The system I have just described should be considered a good foundation for a more extensive command language processor. It is fundamental in its concepts but contains no provision for such things as echo control of the command files as they are processed, user terminal input for selected commands, and intermediate error control or abort. The system I have been using is more advanced than the basic one described here but the main foundation I used is exactly as described above.

Another consideration is that of parameter inputs to the actual programs that are executed from a command file. In the above example all of the called programs derived their parameters directly from the command line itself. In many cases, however, the program may output a question and expect some response from the user terminal. If the program accepts this response via the monitor KBD subroutine, the response can also be part of the command file just as the commands are. The method for making monitor subroutines such as KBD available to the user programs was the topic for the second article in this series (*Kilobaud #2*).

In summary, we have developed here the basis for a system which allows user commands to be written in an easily interpreted manner and also one that allows new commands to be added to the system with no modification to the monitor itself. The ideas are basic but expandable, limited only by your imagination (and perhaps memory, unfortunately). The inclusion of this system, or any similar system which is as flexible, will prove to be an invaluable addition to the *expanding hobbyist's* operating system. ■

Howard R. Bendrot
The Computer Mart of New Jersey
501 Route 27
Iselin NJ 08830

The Slow-Stepping Debugger

Anyone doing machine level programming or hardware development can appreciate the convenience of a single-step switch. Single-stepping through a program while watching the address and data lights is a significant aid in locating where the program or hardware dies. However, nothing can be more monotonous than repeatedly pushing the single-step switch while executing an endless loop program.

Ever since the announcement that the new Altair 8800B had, in addition to the single-step function, a slow-step function, I have been envious of that and regretted that my Imsai 8080 did not have such a feature. Slow-stepping will allow you to rapidly single-step (or slowly run if you prefer) through a program by simply holding a button depressed, thus saving much wear and tear on single-step switches, fingers and patience.

The Original Schematic

Upon examination of the front panel schematic for the Imsai, I discovered that all of the components for this function were present and that with minor modification I could implement both single and slow step.

Fig. 1 shows a portion of the front panel schematic as it appears in the Imsai user manual. Lifting or depressing the single-step switch while in the not run mode causes U17 (74123), a one-shot, to fire for approximately 1.5 ms. The high-to-low transistion of the $\overline{Q}$ signal from Pin 4 sets flip-flop U19 and in essence, allows the processor to run. However, during the next machine cycle, the PSYNC signal will reset U19, causing the processor to suspend operation — thus completing one step. The primary purpose of the one-shot (U17) is to clean up the switch contact closure and provide a clean clock signal to U19.

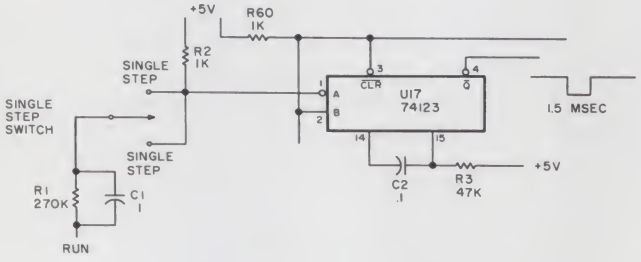


Fig. 1. Original single-step circuit.

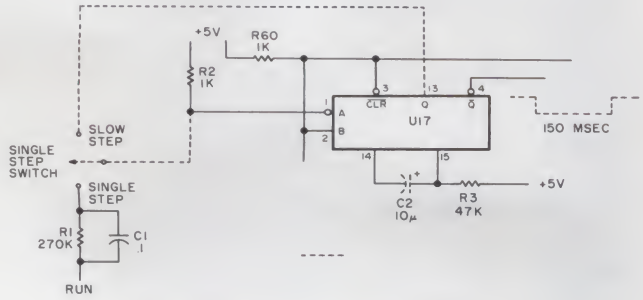


Fig. 2. Modified single-slow-step circuit.

The Modification

By rewiring the circuit and changing the value of C2 from .1uF to 10 uF (as shown in Fig. 2) U17's pulse width is now approximately 150 ms. When the single-step switch is depressed, a normal single-step cycle is generated. However, by lifting the switch, the Q output (Pin 13) is now returned to the input (Pin 1), causing the circuit to operate as a bistable multivibrator with a repetition rate of approximately 6 Hz.

Generally, it is not a good engineering practice to cause a race condition by tying the one-shot output directly to the input. However, the 74123 has sufficient propagation delay to allow this to work reliably. One word of caution — the output pulse width for retriggering is extremely narrow and may be hard to see on all but a laboratory quality oscilloscope. I have modified several front panels and have tried many different 74123s and have yet to find any unreliable.

The modification is most easily made if you have not already assembled your front panel. If you have, the single-step switch must be desoldered and removed to facilitate the cutting of traces on the component side of the board. Cut all three traces going to the switch solder pads on the component side of the front panel board as shown in Fig. 3.

On the trace side of the CPA cut all traces going to the solder pads of S2, the single-step switch, and add the wiring shown in Fig. 4. Capacitor C2 is located near the top right hand corner of the CPA board and must be changed from a .1uF disk capacitor to a 10uF tantalum capacitor. Observe that the positive lead of the capacitor must go up to the junction of R3 and U17 (Pin 15).

The total modification should take no more than 30 minutes to perform and cost no more than the price of a good 10uF capacitor.■

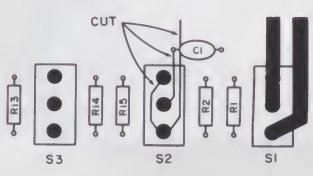


Fig. 3. CPA front side.

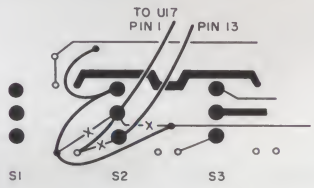


Fig. 4. CPA trace side.

Apple Introduces the First Low Cost Microcomputer System with a Video Terminal and 8K Bytes of RAM on a Single PC Card.

The Apple Computer. A truly complete microcomputer system on a single PC board. Based on the MOS Technology 6502 microprocessor, the Apple also has a built-in video terminal and sockets for 8K bytes of on-board RAM memory. With the addition of a keyboard and video monitor, you'll have an extremely powerful computer system that can be used for anything from developing programs to playing games or running BASIC.

Combining the computer, video terminal and dynamic memory on a single board has resulted in a large reduction in chip count, which means more reliability and lowered cost. Since the Apple comes fully assembled, tested & burned-in and has a complete power supply on-board, initial set-up is essentially "hassle free" and you can be running within minutes. At $666.66 (including 4K bytes RAM!) it opens many new possibilities for users and systems manufacturers.

You Don't Need an Expensive Teletype.

Using the built-in video terminal and keyboard interface, you avoid all the expense, noise and maintenance associated with a teletype. And the Apple video terminal is six times faster than a teletype, which means more throughput and less waiting. The Apple connects directly to a video monitor (or home TV with an inexpensive RF modulator) and displays 960 easy to read characters in 24 rows of 40 characters per line with automatic scrolling. The video display section contains its own 1K bytes of memory, so all the RAM memory is available for user programs. And the Keyboard Interface lets you use almost any ASCII-encoded keyboard.

The Apple Computer makes it possible for many people with limited budgets to step up to a video terminal as an I/O device for their computer.

No More Switches, No More Lights.

Compared to switches and LED's, a video terminal can display vast amounts of information simultaneously. The Apple video terminal can display the contents of 192 memory locations at once on the screen. And the firmware in PROMS enables you to enter, display and debug programs (all in hex) from the keyboard, rendering a front panel unnecessary. The firmware also allows your programs to print characters on the display, and since you'll be looking at letters and numbers instead of just LED's, the door is open to all kinds of alphanumeric software (i.e., Games and BASIC).

8K Bytes RAM in 16 Chips!

The Apple Computer uses the new 16-pin 4K dynamic memory chips. They are faster and take ¼ the space and power of even the low power 2102's (the memory chip that everyone else uses). That means 8K bytes in sixteen chips. It also means no more 28 amp power supplies.

The system is fully expandable to 65K via an edge connector which carries both the address and data busses, power supplies and all timing signals. All dynamic memory refreshing for both on and off-board memory is done automatically. Also, the Apple Computer can be upgraded to use the 16K chips when they become available. That's 32K bytes on-board RAM in 16 IC's—the equivalent of 256 2102's!

A Little Cassette Board That Works!

Unlike many other cassette boards on the marketplace, ours works every time. It plugs directly into the upright connector on the main board and stands only 2" tall. And since it is very fast (1500 bits per second), you can read or write 4K bytes in about 20 seconds. All timing is done in software, which results in crystal-controlled accuracy and uniformity from unit to unit.

Unlike some other cassette interfaces which require an expensive tape recorder, the Apple Cassette Interface works reliably with almost any audio-grade cassette recorder.

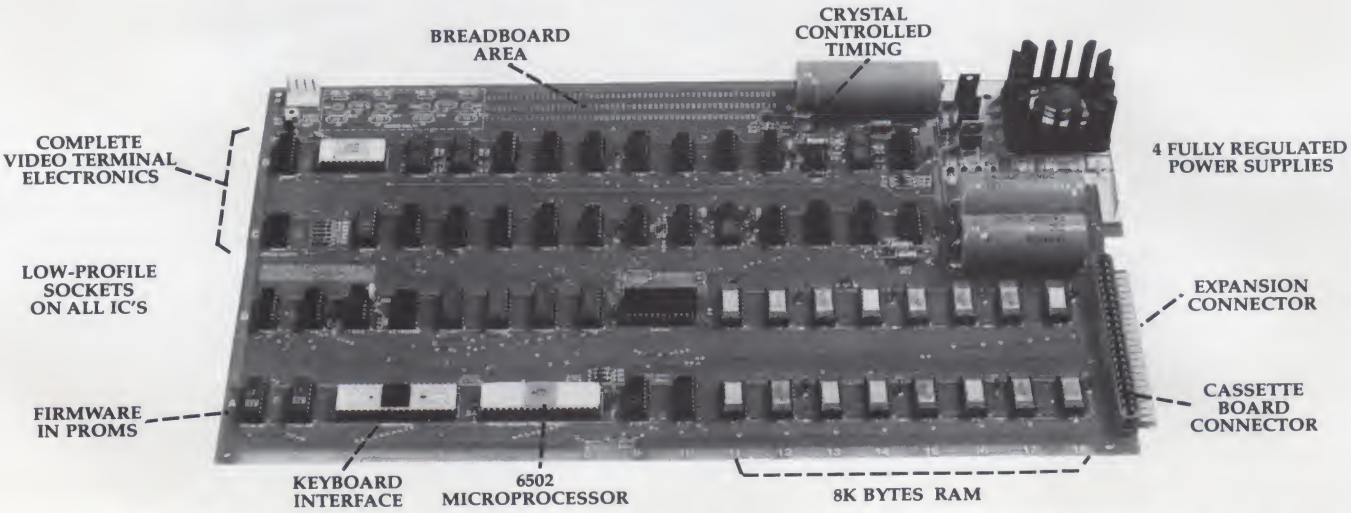
Software:

A tape of **APPLE BASIC** is included free with the Cassette Interface. Apple Basic features immediate error messages and fast execution, and lets you program in a higher level language immediately and without added cost. Also available **now** are a dis-assembler and many games, with many software packages, (including a macro assembler) in the works. And since our philosophy is to provide software for our machines free or at minimal cost, you won't be continually paying for access to this growing software library.

The Apple Computer is in stock at almost all major computer stores. (If your local computer store doesn't carry our products, encourage them or write us direct). **Dealer inquiries invited.**

Byte into an Apple $666.66*

* includes 4K bytes RAM



APPLE Computer Company • 770 Welch Rd., Palo Alto, CA 94304 • (415) 326-4248

Have you written Software for your Altair™ Computer?

The Altair 8800 computer was the first micro produced for the general public and remains number one in sales, with more than 8,000 mainframes in the field. The wide acceptance of the Altair computer and its rapid adaptation to many diversified applications has truly turned the dream of the affordable computer into a reality.

Yet the machine itself, remarkable as it is, represents only the beginning. The right Software, tailored to meet a user's specific requirements, is a vital part of any computer system. MITS wants to insure that Altair users everywhere have the best applications software available today and in the future. For this reason, a new MITS subsidiary, the ALTAIR SOFTWARE DISTRIBUTION COMPANY, has been formed. Its purpose: to acquire the highest quality software possible and distribute it nationally through Altair Computer Centers.

That's where you come in. The ASDC will pay substantial royalties to the originators of all software accepted into the ASDC library. If you have written business, industrial or commercial use software for the Altair 8800, ASDC wants to hear from you. It is the aim of the ASDC to stimulate and reward creativity in producing useful software that makes those dreams of "computers for everyone" come true. The ASDC will select only software that measures up to its high standards for system design, coding and documentation. The software will then be further documented and distributed through Altair Computer Centers around the country.

For more information on how to submit software to the ASDC, ask your Local Altair Computer Center for an ASDC Software Submittal Packet or contact the ALTAIR SOFTWARE DISTRIBUTION COMPANY.

A subsidiary of MITS

ASDC

ALTAIR SOFTWARE DISTRIBUTION COMPANY

3330 Peachtree Road, Suite 343 Atlanta, Georgia 30326 404-231-2308

M-4

see next page for a listing of Altair Computer Centers

ALTAIR COMPUTER CENTERS

BEAVERTON, OR 97005
8105 SW Nimbus Ave.
(503)-644-2314

BERKELEY, CA 94710
1044 University Ave.
(415)-845-5300

SANTA MONICA, CA 90401
820 Broadway
(213)-451-0713

DENVER, CO 80211
2839 W. 44th Ave.
(303)-458-5444

ALBUQUERQUE, NM 87110
3120 San Mateo N.E.
(505)-883-8282;883-8283

TUCSON, AZ 85711
4941 East 29th St.
(602)-748-7363

LINCOLN, NB 68503
611 N. 27th St.
Suite 9
(402)-747-2800

LITTLE ROCK, AR 72206
2412 Broadway
(501)-371-0449

TULSA, OK 74135
5345 East Forty First St.
110 The Annex
(918)-664-4564

HOUSTON, TX 77036
5750 Bintliff Drive
(713)-780-8981

RICHMOND, VA 23230
4503 West Broad St.
(804)-335-5773

SPRINGFIELD, VA 22150
6605A Backlick Rd.
(703)-569-1110

CHARLESTON, W. VA. 25301
Municipal Parking Building
Suite 5
(304)-345-1360

EAGAN, MN 55122
3938 Beau D'Rue Drive
(612)-452-2567

ANN ARBOR, MI 48104
310 East Washington Street
(313)-995-7616

WINDSOR LOCKS, CT 06096
63 South Main Street
(203)-627-0188

PARK RIDGE, IL 60068
517 Talcott Rd.
(312)-823-2388

ST. LOUIS, MO 63130
8123-25 Page Blvd.
(314)-427-6116

NASHVILLE, TN 37203
1600 Hayes St.
Suite 103
(615)-329-1979

BURLINGTON, MA 01803
120 Cambridge St.
(617)-272-8770

ALBANY, NY 12211
269 Osborne Road
(518)-459-6140

NEW YORK, NY 10018
55 West 39th St.
(212)-221-1404

ATLANTA, GA 30305
3330 Piedmont Road
(404)-231-1691

TAMPA, FL 33614
5405 B Southern Comfort Blvd.
(813)-886-9890

ASDC

3330 Peachtree Road, Suite 343
Atlanta, Georgia 30326

Gabriel F. Gargiulo
17 Whitney St.
East Hartford CT 06118

BASIC – The Easy Way

You have already learned the simplest parts of the BASIC computer language and are able to use them easily. At this point you want to know more about the language's capabilities. Since BASIC is designed to be simple, there is not very much more to it. There are some things that give beginners a hard time however. These are worth talking about.

LET can be used to solve complex arithmetic problems. Suppose you want to add 5 to the variable B1. Since BASIC automatically puts zeros into all numeric variables at the start, you can assume B1 has zeros in it. You add 5 to B1 by saying:

```
100 LET B1 = B1 + 5
```

This means let B1 equal the sum of B1 plus 5. BASIC looks at the right side of the assignment *equation*, adds B1 and 5, and puts the answer *into the variable* on the left of the equation. So 5 and 0 are added and the result is put into B1. Now if you execute the statement again, B1 has a 5 in it, so 5 and 5 are added and the result goes into B1. You could put the answer into any other numeric variable, such as Q5.

```
100 LET Q5 = B1 + 5
```

This means add up B1 and 5 and put the answer into Q5. The left side of the equation must contain a variable and only a variable. You cannot say LET 5 = B1 + 5. On the right side, however, you can do any arithmetic operation you want. You can add, subtract, multiply, divide, and combinations of these.

To subtract, you use a "-". So B1 minus 5 is B1 - 5. For example, to subtract 5 from B1 you could say:

```
100 B1 = B1 - 5
```

To multiply, you use *. If you remember your high school algebra, you used a raised dot to indicate multiplication. There is no raised dot on computer keyboards, * is used instead. To multiply B1 times 5, you would say:

```
100 B1 = B1 * 5
```

In algebra you could just put two letters next to each other to indicate multiplication. You cannot do this in any programming language, let alone BASIC. AB meant A times B. In BASIC you have to say A * B. Division is like algebra. One divided by 2 is 1/2. One half of B1 is B1 / 2. To divide B1 by 2 say:

```
100 B1 = B1 / 2
```

Exponentiation was done in algebra by writing the exponent a bit higher and to the right of the number. There is no way to do this on any computer. Instead you put two asterisks in front of the exponent. (On some computers you use an arrow pointing up.) So B1 squared is B1 **2 or B1 ↑ 2. To square B1 you say:

```
100 LET B1 = B1 ** 2
```

Now we are ready to start combining arithmetic operations. How about a useful example? Let's solve the formula for area of circle. A = pi R^2. This means square R and multiply the product by pi. R squared is R**2 (or R*R, since this is easier than squaring). To multiply the product by pi, say 3.14159 * R * R, or

```
100 LET A = 3.14159 * R * R
```

Please note, it does not matter whether your computer multiplies R times R then multiplies the result by 3.14159, or whether it multiplies 3.14159 times R first, then the result by R. The answer is the same.

There are times when it does matter whether the computer does one operation before another. In these cases, you can use parentheses to tell the machine what operations to do in what order. There is a hierarchy of types of operation. At the top is exponentiation, then multiplication and division together, then at the bottom, addition and subtraction. Normally the computer does exponentiation first, then multiplication, then addition and subtraction. So if you say

```
100 LET B = 3 + A * B
```

it multiplies A and B first, then adds 3 to the answer. If this is not what you wanted, you can say

```
100 LET B = (3+A) *B
```

Then it will add 3 and A and multiply the sum by B. Parentheses can be used to tell the computer what order to do things in. When you put parentheses around some numbers or variables, it means they should be taken as a whole. This is just like in algebra. Whenever you have any doubt about the order in which operations will be performed, use parentheses. You may use parentheses even when you don't need them. They can make an expression easier to under-

stand.

BASIC is capable of showing only six significant digits. (Some versions can show more.) This means that some digits will be lost if you have more than six digits in a number that you enter or that turns out as the result of some computation. If you have a decimal fraction, you will only lose the least significant digits. If your answer is 2.3814532, BASIC will drop the 32 and keep the 2.38145 since it can handle only six significant digits. What if 1,000,000 were the result of an assignment? There would be more than six digits. BASIC would have to drop off some digits. But it can't drop off the last zero because it would be dividing by ten. It can't drop the 1, since only zeros would be left. So it reverts to *scientific notation*. It expresses the number as a decimal fraction (for example .1) multiplied by a power of ten. The power of ten is shown as E followed by the power to which ten is raised. Ten to the second power is E2. (100, since 10 times 10 is 100.) Therefore .1 E2 means .1 times 10 to the 2nd power: .1 times 100, that is 10. The first power of 10 is 10. So .1 E1 is .1 times 10 or 1.

Negative exponents work too. Ten to the minus 1 power is .1. So .1E-1 is .1 times .1 or .01. Ten to the zero power is 1, so .1E0 is .1. Note what happens to the decimal point in the result as you change the power of 10 (see Table 1).

It is easier to understand negative exponents as meaning "shift the decimal point *that* many places to the left" and positive (or unsigned) exponents as meaning "shift *that* many places to the right."

The .1 E1 means "shift the decimal point 1 place to the right," which gives a result of 1. The .1E-1 means "shift one decimal place to the left," which results in .01. An exponent of 0 would mean "shift *no* places to the right or left." A few more examples: .301E-2 means .00301, and .786 E5 means 78600.

You can also enter numbers in this notation. BASIC sees E and recognizes the number as being in scientific notation. You can put in the sign if you want, for example: E+2 E-5, or leave it out: E5. (Then BASIC assumes you meant E + 5.) You can show the exponent in 1 or 2 digits. So E5, E+5 and E+05 are all equivalent.

The PRINT imperative is the one you use when you want something to appear on your video screen or typewriter. If you say PRINT "LITERAL" the word LITERAL will appear on the screen. If you say PRINT V the value of V will be put on the screen. If you want to print two or more variables or literals next to each other, you just join them with a semicolon or a comma. PRINT "RATE = "; R. This will print RATE = 5 if R contains a 5.

If you use semicolons the items (literals or variables) are printed next to each other:

```
100 PRINT "A"; "B"; "C"
```

will print

```
ABC
```

If you are using commas it works differently. BASIC divides the screen or typewriter into 5 zones. If you said PRINT "RATE =", BASIC will put "RATE =" into zone 1, and the value of R in zone 2, giving, for example:

```
RATE = 5
```

If you have 5 variables or literals, they will be printed across the screen or typewriter.

```
100 PRINT "**", "RATE1 =", R1, "RATE2 =", R2
```

would give, (if R1 is 5 and R2 is 6), the following:

```
** RATE1= 5 RATE2 = 6
```

You can mix commas and semicolons. Then you get a combination of the above. There is an unusual case where you leave a *dangling* comma or semicolon.

```
100 PRINT "RATE =";
```

When the program reaches this statement *nothing* is printed. To get some output you have to hit another PRINT statement. Then the output of the second comes out next to the output of the first.
For example:

```
100 PRINT "RATE =";
110 R1 = 15 * (A2 + 5)
120 PRINT R1
```

might give you:

```
RATE = 300
```

Another interesting thing you can do with PRINT: You can use an expression instead of a variable or literal.

```
100 PRINT "RATE = "; 15 * (A2 + 5).
```

This gives the same result as the preceding example.

IF. You can compare two things in BASIC and go to a statement number, depending on the outcome. The IF statement is very simple, unlike in COBOL where it can be very complex. It is more like assembler language, where each comparison is simple. You compare a variable to a literal, another variable, or an expression. The comparison can be greater than (symbol >), less than (symbol <), equal to (symbol =), not equal to (symbol <>), less than or equal to (symbol =<), greater than or equal to (symbol =>). You must compare like types, numeric to numeric, and string to string.
To ask if R1 equals 5, say:

```
100 IF R1 = 5 . . .
```

If R1 is not equal to 5, say:

```
100 IF R1 <>5 . . .
```

IF allows you to go to a statement number if the outcome of the comparison is true.

```
100 IF R1 = 5 THEN 120
```

means go to statement 120 if R1 equals 5.

Since you can't compare unlike types

```
100 IF R$ = 5 THEN 200
```

is illegal, but

```
100 IF R$ = "5" THEN 200
```

is legal.

BASIC's *functions* (see Table 2) allow you to do things that would be difficult to program. These functions require little explanation. The x stands for any numeric variable. In BASIC's functions, the x can be any variable. RND is different. You can say RND (0). Then you get a random number betwen 0 and 1, but each time you run the program you will get the same series of numbers. RND (x) gives you different results, depending on whether x is positive or negative. If x is positive, you get a series of numbers that are the same whenever x is the same. Putting a 3 in x will give you one series. Every time you put a 3 in x you get the same series. Putting a 5 in x will give you another series. If x is negative you get an unpredictable series of numbers.

GOSUB. Sometimes it is convenient to put a part of your program into a *subroutine* and use it from any other part of your program. Say you have a subroutine whose first statement is 900. You

Number in scientific notation	Value
.1 E 3	100.
.1 E 2	10.
.1 E 1	1.
.1 E 0	.1
.1 E-1	.01
.1 E-2	.001
.1 E-3	.0001

Table 1

want to use it in your program at statement 200.

```
200 GOSUB 900
```

At statement 200 the program goes to statement 900. This looks suspiciously like a GO. It is similar, but it has one important difference — the subroutine must end with a RETURN. The RETURN sends the program back to the statement right after 200. This is useful because you may want to use the subroutine from two or more parts of your program.

```
200 GOSUB 900
210 PRINT "BACK FROM SUB"
400 GO SUB 900
410 PRINT "BACK FROM SUB AGAIN"
420 END
900 PRINT "START SUB 900"
910 PRINT "READY TO RETURN"
920 RETURN
```

At statement 200 the program goes to 900. In the subroutine there is a return which sends the program back to 210. Later, at statement 400, the program also goes to 900. This time, however, the *return* sends back to 410. Every subroutine that you go to with GOSUB must end with a RETURN. Any subroutine may have a GOSUB in it.

```
100 GOSUB 700
700 GOSUB 800
710 RETURN
800 REMARK SUB 800 STARTS
HERE
810 RETURN
```

The rule is, when a RETURN is encountered, control goes back to the most recent GOSUB.

Multiple statements per line. Some versions of BASIC allow you to have more than one imperative on a line. They are separated by colons.

```
100 FOR I = 1 TO 10: PRINT I: NEXT I
```

TAB allows you to space the printer over before printing. PRINT TAB (10); "COLUM 11" prints "column 11" beginning in column 11. You can use a variable or an expression as well.

```
PRINT TAB (A * B); A$
```

STOP, when executed, makes the program halt. BASIC then asks you if you want the program to start again. If you reply in the affirmative, the program continues as if nothing had happened. STOP can be used to display a message when it stops. STOP "PROGRAM STOPPED" will stop the program and display "PROGRAM STOPPED".

All the above BASIC statements are statements that can appear in a program. There is another class of BASIC imperatives which cannot appear in a program. They are *commands* (see Table 3). They tell BASIC what to do with your program. They cannot have statement numbers in front of them.

You should now be familiar with all the elements of the BASIC language that are common to all computers. Now you should use it. That is really the only way to become thoroughly familiar with it. Write programs. Long programs and short programs. Write a program to balance your checking account. Use BASIC to do simple calculations that you can do on a pocket calculator. Use it every day and in a short time you will become an expert in this powerful language.■

ABS (x) gives absolute value of x, i.e., drop the sign if negative.
INT (x) drop the decimal fraction, if any, from x.
SGN (x) gives +1 if x is positive, 0 if zero, -1 if negative.
SIN (x) The sine of the angle x (expressed in radians).
COS (x) The cosine of the angle x (expressed in radians).
TAN (x) The tangent of the angle x (expressed in radians).
ATN (x) The arctangent of the angle x (expressed in radians).
SQR (x) The square root of x.
EXP (x) The value of E to the power of x, where E = 2.71828.
LOG (x) The natural log of x.

Table 2

LIST: If you want to look at statements in your program, use LIST. LIST means display the whole program on the screen or terminal. To list part of your program, type LIST starting-line-no end-line-no. LIST 100,300 lists statements from 100 to 300. If there is no statement 100, you will see the first line number greater than 100. If there is no statement 300, you will see up to the statement number greater than 300. On some machines you say LIST 100-300.
RUN: To get your program to start executing and actually do what the statements tell the computer to do you type in RUN. Then BASIC puts the program into executable form and gives control to it. If however it finds an inconsistency, such as GO TO a line number that doesn't exist, the error is displayed on the screen and execution never starts.
RESEQUENCE: BASIC can assign new numbers to a program's statements. It will give the first statement line number 100, and each line number will be 10 higher than the previous. If you don't like these numbers, you can have BASIC number them differently. RESEQUENCE 50,5 means "start at line 50, and increment by 5." On some computers you say RENUMBER.
SAVE: When you type in a program it goes into core storage or main memory. If you turn off your computer, the program will be gone forever. SAVE stores the program on external storage, usually a disk.

Table 3

OUR COMPUTER MAKES MUSIC!

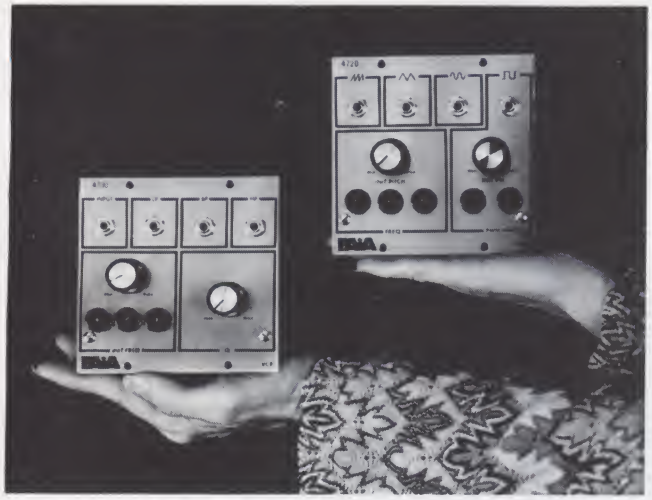
HERE'S HOW:

EQUALLY TEMPERED DIGITAL TO ANALOG CONVERTER

Unlike more conventional R-2R ladder type digital to analog converters, the PAIA 8780 kit is based on a multiplying principle that allows the module to generate the exact exponential stair-step function required to make even the simplest linear response oscillators and filters produce equally tempered musical intervals. The 8780 uses only six bits of data to generate over 5 octaves of control voltage. In an 8 bit system, the remaining 2 bits are ordinarily reserved for trigger flags, but may be used to extend the range of the converter or provide micro-tonal tunings.

The module is physically and electrically compatible with the complete line of PAIA music synthesizer modules and is easily interfaced to any micro-processor with or without hand-shaking logic.

#8780 D/A CONVERTER Kit..$34.95 (plus $1.00 postage)



MUSIC SYNTHESIZER MODULES

PAIA offers a complete line of low-cost voltage controlled music synthesizer module kits including the 4720 Oscillator and 4730 filter shown. Both units feature linear freq./control voltage response and 16 Hz. to 16 kHz. range.

The 4720 VCO produces ramp, triangle, sine and pulse waveforms. The companion 4730 VCF is a state variable design with simultaneously available low-pass, band-pass and high-pass outputs, all with "Q" adjustable from .5 to 150.

Other modules in the series include: Voltage Controlled Amplifiers, Balanced Modulators, Envelope Generators, Reverb Units, Noise Sources and Power Supplied.

All modules are compatible with the PAIA 8780 Equally Tempered DAC for easy computer/microprocessor/micro-controller interface.

#4720 VCO Kit..... $34.95 .....(plus $1.00 postage)
#4730 VCF Kit..... $37.95..... (plus $1.00 postage)

COMPUTER TECHNOLOGY APPLIED.....
....IN A SELF-CONTAINED PROGRAMMABLE DRUM SET

While most electronic rhythm units offer only a limited choice of pre-determined rhythm patterns, the PAIA Programmable Drum Set allows the user to tailor pattern, time signature and drum sounds to each application. Among the unique features provided by the unit are touch sensitive electronic controls and the provision for an independently structured bridge rhythm.

Battery powered, the drum set includes a "memory save" switch which provides a lowered "keep-alive" voltage to the drum set's 256 byte memory.

#3750 PROGRAMMABLE DRUM SET Kit $79.95
(plus $3.00 shipping & insurance)

All these and more in our FREE CATALOG.....

FROM THE INNOVATORS AT:

PAIA ELECTRONICS

DEPT 4-K
1020 W. WILSHIRE BLVD.
OKLAHOMA CITY, OK 73116

The Programmable Drum Set and all Synthesizer Modules are designed to play through any hi-fi or musical instrument amplifier.

Tim Barry
P.O. Box 43
Mountain View CA 94043

Now You Can Use

I've written a few assembly-language timing loops in my day, and I probably used the same approach a lot of other people have . . . trial and error. I'm through doing it that way and I suspect you will be too after reading Tim's article. He not only does a good job of covering the subject of timing and timing loops but he also mentions some of the sneaky pitfalls waiting when we go from RAM to PROM with certain programs. — John.

In the course of developing programs you will occasionally need to concern yourself with how fast a program section executes. The most common case is when you need to insert a fixed time delay into the program to adjust its execution rate to match the rate of a slower peripheral. Another common case is when you need to see if some particular program section is causing system speed problems. These and other time critical program sections require careful analysis to ensure correct operation.

In this article we will examine the various elements that affect the execution speed of individual computer instructions. We will then look at the general equations which allow us to either compute the execution time of a given loop or to develop a loop which will delay for an exact period of time. Finally, we will discuss some of the pitfalls to avoid when using software delays in your programs. Throughout the article we will use real microcomputer timing examples based on the 8080 microprocessor.

Computing Instruction Execution Times

The amount of time required for a computer to execute any given instruction is a function of the CPU clock cycle speed, the speed of the memory where the instruction is stored, the speed of the memory (if any) operated upon by the instruction, and the actual instruction itself. To understand how these combine to form an instruction execution time we need to step back a bit and discuss how computers execute instructions.

Each instruction executed by the computer is fetched from the memory and decoded. If the instruction requires other operands from memory, these are also fetched. Once the operands are in place the instruction is executed and the result is either ignored, left in a register or transferred back into the memory. This cycle of transferring instructions and data to and from memory continues as each instruction in the program is executed.

Variations in instruction speed arise from three factors: clock frequency, the number of memory accesses required to transfer operands between the CPU and memory, and the number of machine states required to execute the instruction. Most functions in the computer are somehow related to the master system clock. Therefore, it stands to reason that the faster the clock, the faster the instructions will execute. Maximum clock frequency is dependent upon the actual CPU hardware and typical microprocessors run with clocks from 1-2 MHz.

Anytime the CPU needs to move data to or from the memory it must perform a memory access. It does this by placing the address of the location to be accessed on the system memory address bus, manipulating the control signals to cause a read or write, and waiting for the memory to signal that the transfer is complete (or not waiting unless the memory says the transfer is not complete). If the memories are as fast as the CPU cycle time, the CPU will not need to slow down while transferring data to or from the memory. However, if the memory cannot keep up, the CPU will have to wait for each transfer to be completed. Since all instructions and most data must be fetched from memory, this can result in significant increases in program execution time.

In microprocessors any required memory waits are usually generated from the system clock. The normal procedure is for a single wait state to be equal to one clock cycle. Thus, a microprocessor running with a 2 MHz clock would have a single clock cycle time of 500 ns. Memories with access times less than 500 ns would require no wait states, memories with access times in the range of 500-1000 ns would require one 500 ns wait, and so on. Commonly used memories require from 0-2 waits per access in a 2 MHz system.

Finally, in addition to memory accesses and clock speed, the number of machine states required for the CPU to execute the instruction affects instruction execution time. A machine state is a period of time during which the computer CPU executes operations among the internal architectural elements. The CPU can only do a limited number of things during a single state, so the more complicated instructions will require more states. A single state is usually equal to one clock cycle, so the more states, the more time required to execute the instruction. In the 8080 microprocessor, instructions take from 4 to 18 states to execute.

Now that we have discussed the basic ingredients we can discuss the actual timing of instructions. The general timing equation for instructions executed from

Software Timing Loops

memory is as shown in Table 3.

The first term, T_0, is derived from the system clock speed. A 2 MHz clock has a $T\emptyset = 1/T\emptyset = 1/(2\ MHz) = 500$ ns. The second term, N*A, accounts for the amount of time spent waiting for the memory. It is derived by multiplying the number of waits required for each access by the number of accesses required. (Remember, all instructions must be fetched from memory; A is always greater than or equal to one.) If the memory is faster than T_0, this term will be zero. The third term, S, is the total number of states required for the CPU to execute the instruction.

For example, consider an instruction which requires two memory accesses and seven machine states running in a system with a 2 MHz clock and memories which require one wait cycle per access. The execution time of this instruction would be as shown in Table 4.

In systems with memories having different access times the equation remains basically the same. The only change is in the wait cycle term. This becomes $N_1A_1+N_2A_2 \ldots N_NA_N$, where the Ns and As are for the number of waits and the number of accesses for each type of memory. The term is seldom used in this full form. It is most often encountered in microcomputer systems where the program is located in ROM (read only memory) and the data is being stored in RAM (read/write memory). In this case the general equation is shown in Table 5.

For example, consider an instruction which executes in 10 states and requires three memory accesses to ROM and one to RAM. In a system with a 2 MHz clock, ROM which requires two waits per access and RAM which requires one wait per access, the computation is shown in Table 6. With a little practice, computing execution times for individual instructions becomes a straightforward job.

8080 Instruction Execution Times

I use the 8080 in a lot of day-to-day work. Tables 1 and 2 list the 8080 instructions in a format which helps me quickly compute execution times for any system.

Table 1 is a summary of the execution characteristics of each instruction. The first column lists the instruction mnemonics. (Some instructions with identical timing characteristics are listed together.) The notation convention used is as shown in Table 7.

The second column lists the number of memory bytes required by the instruction. The third column lists the total number of machine states required to execute the instruction.

The last three columns are used to determine the number and type of memory

$$T_E = T\emptyset*(N*A+S)$$

where $T\emptyset$ = clock cycle time
N = number of wait states/access
A = number of memory accesses
S = total number of machine states

Table 3.

$T_E = T\emptyset*(N*A+S)$
$T\emptyset = 500\ ns,\ N = 1,\ A = 2,\ S = 7$
$T_E = 500\ ns*(1*2+7)$
$= 500\ ns*(9)$
$= 4.5\ usec$

Table 4.

$$T_E = T\emptyset*(N_{RAM}*A_{RAM}+N_{ROM}*A_{ROM}+S)$$

where $T\emptyset$ = clock cycle time
N_{RAM} = number of wait states per RAM access
A_{RAM} = number of RAM accesses
N_{ROM} = number of wait states per ROM access
A_{ROM} = number of ROM accesses

Table 5.

$T_E = T\emptyset*(N_{RAM}*A_{RAM}+N_{ROM}*A_{ROM}+S)$
$T\emptyset = 500\ ns,\ N_{RAM} = 1,\ A_{RAM} = 1,\ N_{ROM} = 2,\ A_{ROM} = 3,\ S = 10$
$T_E = 500\ ns*(1*1+2*3+10)$
$= 500\ ns*16$
$= 8\ usec$

Table 6.

INSTRUCTION	BYTES	STATES	TOTAL ACCESS	RAM ACCESS	ROM ACCESS
MOV D,S	1	5	1	0	1
MOV M,S/MOV D,M	1	7	2	1	1
MVI D,D8	2	7	2	0	2
MVI M,D8	2	10	3	1	2
LXI RP,D16	3	10	3	0	3
LDA ADDR/STA ADDR	3	13	4	1	3
LHLD ADDR/SHLD ADDR	3	16	5	2	3
LDAX RD/STAX RD	1	7	2	1	1
PUSH RS	1	11	3	2	1
POP RS	1	10	3	2	1
PCHL/SPHL	1	5	1	0	1
XCHG	1	4	1	0	1
XTHL	1	18	5	4	1
IN IO/OUT IO	2	10	2	0	2
STC/CMC/STC/DAA	1	4	1	0	1
RAL/RAR/RLC/RRC	1	4	1	0	1
A@S	1	4	1	0	1
A@M	1	7	2	1	1
A@D8	2	7	2	0	2
DAD RP	1	10	1	0	1
INR S/DCR/S	1	5	1	0	1
INR M/DCR/M	1	10	3	2	1
INX RP/DCX RP	1	5	1	0	1
JMP ADDR	3	10	3	0	3
J(COND) ADDR	3	10	3	0	3
CALL ADDR	3	17	5	2	3
C(COND) ADDR*	3	11/17	3/5	0/2	3/3
RET	1	10	3	2	1
R(COND)	1	5/11	1/3	0/2	1/1
RST L	1	11	3	2	1
NOP	1	4	1	0	1
EI/DI	1	4	1	0	1
HLT	1	7	1	0	1

**First number = execution time if condition is false.*

Table 1. 8080 instruction execution characteristics.

			Memory States						
			Single Memory Speed			Wait	Split Memory Speed		
INSTRUCTION	BYTES	STATES	0	1	2	0/1	0/2	1/0	1/2
MOV D,S	1	5	2.5	3	3.5	3	3.5	2.5	3.5
MOV M,S/MOV D,M	1	7	3.5	4.5	5.5	4	4.5	4	5
MVI D,D8	2	7	3.5	4.5	5.5	4.5	5.5	3.5	5.5
MVI M,D8	2	10	5	6.5	8	6	7	5.5	7.5
LXI RP,D16	3	10	5	6.5	8	6.5	8	5	8
LDA ADDR/STA ADDR	3	13	6.5	8.5	10.5	8	9.5	7	10
LHLD ADDR/SHLD ADDR	3	16	8	10.5	13	9.5	11	9	12
LDAX RD/STAX RD	1	7	3.5	4.5	5.5	4	4.5	4	5
PUSH RS	1	11	5.5	7	8.5	6	6.5	6.5	7.5
POP RS	1	10	5	6.5	8	5.5	6	6	7
PCHL/SPHL	1	5	2.5	3	3.5	3	3.5	2.5	3.5
XCHG	1	4	2	2.5	3	2.5	3	2	3
XTHL	1	18	9	11.5	14	9.5	10	11	12
IN IO/OUT IO	2	10	5	6.5	8	6.5	8	5	8
STC/CMC/STC/DAA	1	4	2	2.5	3	2.5	3	2	3
RAL/RAR/RLC/RRC	1	4	2	2.5	3	2.5	3	2	3
A@S	1	4	2	2.5	3	2.5	3	2	3
A@M	1	7	3.5	4.5	5.5	4	4.5	4	5
A@D8	2	7	3.5	4.5	5.5	4.5	5.5	3.5	5.5
DAD RP	1	10	5	5.5	6	5.5	6	5	6
INR S/DCR S	1	5	2.5	3	3.5	3	3.5	2.5	3.5
INR M/DCR M	1	10	5	6.5	8	5.5	6	6	7
INX RP/DCX RP	1	5	2.5	3	3.5	3	3.5	2.5	3.5
JMP ADDR	3	10	5	6.5	8	6.5	8	5	8
J(COND) ADDR	3	10	5	6.5	8	6.5	8	5	8
CALL ADDR	3	17	8.5	11	13.5	10	11.5	9.5	12.5
C(COND) ADDR	3	11/17	5.5/8.5	7/11	8.5/13.5	7/10	8.5/11.5	5.5/9.5	8.5/12.5
RET	1	10	5	6.5	8	5.5	6	6	7
R(COND)	1	5/11	2.5/5.5	3/7	3.5/8.5	3/6	3.5/6.5	2.5/6.5	3.5/7.5
RST L	1	11	5.5	7	8.5	6	6.5	6.5	7.5
NOP	1	4	2	2.5	3	2.5	3	2	3
EI/DI	1	4	2	2.5	3	2.5	3	2	3
HLT	1	7	3.5	4	4.5	4	4.5	3.5	4.5

Table 2. 8080 instruction execution times at 2 MHz.

Notation	Meaning
D or S	8-bit 8080 register A, B, C, D, E, H, or L (D = Destination and S = Source)
M	Contents of memory addressed by HL register pair
RP	Register Pair BC, DE, HL, or the stack pointer
RS	Register Pair BC, DE, HL, or program status word
RD	Register Pair BC or DE
ADDR	16-bit memory address
D8	8-bit immediate data
D16	16-bit immediate data
IO	8-bit I/O address
COND	One of eight conditional execution codes
@	Arithmetic/logic operation
L	Integer in range 0-7

Table 7.

$T_E = T_\emptyset * (N*A+S)$
$T_\emptyset$ = 500 ns, N = 0, A = 5, S = 16
T_E = 500 ns*(0+16)
= 8 usec

Table 8.

accesses which take place during program execution. The total access column lists the total number of memory accesses required by the instruction. If your program is loaded into memory with constant access times, you use this number in your execution time computations.

The last two columns are used if your program is loaded in ROM but using RAM for some operations. The ROM Access column lists the number of fetches the processor will execute while transferring the instruction from ROM. This becomes A_{ROM} in the timing equation and it always equals the number of bytes in the instruction. The RAM Access column lists the number of accesses the instruction makes to RAM while transferring operands. This number becomes A_{RAM} in the equation.

For example, consider the LHLD instruction. It requires 18 machine states to load the HL register pair with the contents of two sequential memory locations specified by a 16-bit address. It requires 5 memory accesses: three for instruction fetch and two for operand transfers. Let's compute the execution time for both same and mixed memory applications.

For the example with all memory accesses taking the same amount of time, let's assume that the system is running at 2 MHz using 450 ns access time 2102As for main memory (typical system). Since everything is going in RAM we use the simplified equation, and since the RAMs are faster than the clock we will not need any waits (see Table 8).

Now let's assume the instruction can load the operands from the RAM but is itself stored in a 1702A EROM with access time of 1300 ns. This EROM is 1300 – 500 = 800 ns slower than the clock, so we will need two wait states for the fetch accesses. The equation now becomes that shown in Table 9.

Almost a 40% difference in execution time! If this instruction happened to be in a critical timing loop debugged in RAM, it could cause the program to malfunction if it were transferred to a ROM.

Table 2 shows the actual execution times for 8080 instructions in systems with a 2 MHz clock and a variety of memory configurations. The single number columns show the execution times for instructions running in systems with all memories having the same access times. The split number columns show the execution times for systems with different speed memories. The first number is the number of waits required by the memory used for read/write accesses.The second number is for the number of waits required by the memory used for read only accesses. Thus ½ would mean the program is stored in memories requiring 2 waits per access and is using memory with one wait per access for data storage. (A single 8½ X 11 8080 notebook reference card which combines Tables 1 and 2 with the 8080 timing information is available. Send $1.25 to: Tim Barry, P.O. Box 43, Mt. View CA 94043.)

Calculating Program Execution Times

Now that we have discussed how to compute execution times for individual instructions we can move on to discuss how to time actual program sections. Since programs are really just collections of individual instructions, this is really very easy. The secret is to always have an instruction timing chart handy whenever you need to time a program section.

Programs can be considered to be either straight line or looped. A straight line program section executes each instruction in the area to be timed once. Therefore, the total execution time is simply the sum of the execution times of the instructions:

$$T_E = \Sigma T_I$$

For example, consider the following 8080 program section:

```
      .
      .
      .
    IN 10H
    ANI 7FH
    MOV C,A
    CALL OUTP
      .
      .
      .
```

The execution time of this section in a system with a 2 MHz clock and memories requiring no waits would be found by looking up the times of the individual instructions in Table 2 and adding them together (see Table 13).

$T_E = T_\emptyset * (N_{RAM}*A_{RAM}+N_{ROM}*A_{ROM}+S)$
$T_\emptyset$ = 500 ns, N_{RAM} = 0, A_{RAM} = 2, N_{ROM} = 2, A_{ROM} = 3, S = 16
T_E = 500 ns*(0*2+2*3+16)
= 500 ns(22)
= 11 us

Table 9.

$T_E = \Sigma T_P + T_L$
$= \Sigma T_P + N*\Sigma T_I$
where ΣT_P = execution time of the peripheral instructions
ΣT_I execution time of loop instructions
N = number of times the loop executes

Table 10.

A loop executes a section of code for a fixed number of times. Therefore, the loop execution time equals the sum of the execution times of the instructions in the loop multiplied by the number of times the loop is executed:

$$T_L = N^*\Sigma T_I$$

Making our previous example into a loop which executes 10 times, we see:

```
        MVI E,10
LOOP:   IN 10H
        ANI 7FH
        MOV C,A
        CALL OUTP
        DCR E
        JNZ LOOP
```

Note that we have to add some instructions to convert our straight line program to a loop. The MVI instruction is outside the loop. It is required to set the loop counter to the required value. This is considered a peripheral instruction. Most loops require peripheral instructions to set up for execution and exit conditions. The DCR and JNZ instructions are inside the loop to provide loop control. For timing purposes they are treated just like all the other loop instructions. The total time required is as shown in Table 10.

For our example, N = 10 and the computation for this formula is shown in Table 11. In this example the peripheral instruction makes very little difference in the total execution time. However, on delay loops which are being set for short times they can become significant and it is usually best to include them in all calculations.

$$\Sigma T_P = T_{MVI} = 3.5 \text{ usec}$$
$$\Sigma T_I = T_{IN}+T_{ANI}+T_{MOV}+T_{CALL}+T_{DCR}+T_{JNZ}$$
$$= 5+3.5+2+8.5+2.5+5$$
$$= 26.5 \text{ usec}$$

$$T_E = \Sigma T_P+N^*\Sigma T_I$$
$$= 3.5+10^*26.5$$
$$= 268.5 \text{ usec}$$

Table 11.

$$T_L = \Sigma T_P+N^*\Sigma T_I$$
$$\Sigma T_P = T_{PUSH}+T_{MVI}+T_{POP}+T_{RET} = 6.5+7+6+6 = 25.5$$
$$\Sigma T_I = T_{DCR}+T_{JNZ} = 5.5+8 = 13.5$$
$$N = 131$$
$$T_L = 25.5+131^*13.5$$
$$= 1.8 \text{ ms}$$

Table 12.

$$T_E = \Sigma T_I$$
$$= T_{IN}+T_{ANI}+T_{MOV}+T_{CALL}$$
$$= 5+3.5+2+8.5$$
$$= 19 \text{ usec}$$

Table 13.

$$\Sigma T_P = T_{PUSH}+T_{MVI}+T_{POP}+T_{RET}$$
$$= 5.5+3.5+5+5$$
$$= 19 \text{ usec}$$

$$\Sigma T_I = T_{DCR}+T_{JNZ}$$
$$= 2.5+5$$
$$= 7.5 \text{ usec}$$

Table 14.

Designing Fixed Time Delay Loops

Suppose you are interfacing a cassette tape reader to your trusty microcomputer. Digging through the documentation you discover that you must design a subroutine which provides a one millisecond timing delay. What do you do? Don't panic, the procedure is quite painless.

First, design the routine as a shell. That is, figure out the loop without worrying about how long it takes to execute. A nice, garden variety 8080 delay loop might be as follows:

```
DELAY: PUSH PSW
       MVI A, CONSTANT
LOOP:  DCR A
       JNZ LOOP
       POP PSW
       RET
```

We will compute the loop constant by rearranging our general timing equation. We already know that for a given program section with a loop,

$$T_E = \Sigma T_P+N^*\Sigma T_I$$

Rearranging to solve for the constant N, we see

$$N = \frac{T_E-\Sigma T_P}{\Sigma T_I}$$

Now we time our loop. Assuming a 2 MHz clock and 2102A main memories requiring no waits, the computation is seen in Table 14. Since we need a 1 msec = 1000 usec delay,

$$N = \frac{1000-19}{7.5}$$

$$= 130.8 \approx 131$$

Thus our real loop would become:

```
DELAY:  PUSH PSW
        MVI A,131
LOOP:   DCR A
        JNZ LOOP
        POP PSW
        RET
```

Adjusting Fixed Time Delay Loops

Well, you've had the cassette system running for a few months now and you decide to move that whole program into an EROM to save loading it each time. 1702s are plentiful and cheap, so you buy a card and some memories. You then send them off to be programmed. Back they come and you're ready for the big trial. Unfortunately, all your tapes resist every effort to load them. What happened? You forgot to adjust your delay loops.

The 1702a requires 2 waits per access. Assuming you kept the no wait RAM, you are now operating from the 0/2 column of the timing table. The program now delays as shown in Table 12.

Your timing loop now takes almost twice as long to execute. Fortunately, all you have to do is recompute N based on the new ΣT_P and ΣT_I. The important thing is to remember to recompute the delays before you waste time, money, and effort programming EROMs, or, perish the thought, real ROMs. Whenever you buy a new program you should always check to see if it has any timing loops. If it does, make sure that they are adjusted to match your system.

Program Delay Pitfalls

Making your program dependent upon the execution time characteristics is a good way to save some hardware (i.e., money) in certain applications. There are some potential pitfalls, however, and if you plan to use software delays, you should be aware of them. Probably the greatest pitfall has already been mentioned: lack of portability. A finely tuned delay loop may work just fine in your system. But when you move it to a different system, or even to a different memory in the same system, you can run into problems. The best plan is to surround any critically timed sections

with lots of comments that spell out exactly what the loop does and what conditions were used to time it. (Clock frequency, memory speeds, etc.) This will make them easy to find when modifications become necessary.

Another problem with software timing loops can occur in systems which use interrupts. When an interrupt occurs, the CPU stops executing whatever it's doing, services the interrupt and returns control to the interrupted program. If an interrupt should hit in the middle of a critical timing loop, the time required to service the interrupt routine will probably cause a timing error. Since this does not happen regularly it is the kind of bug that really makes you tear your hair. The only solution is to disable interrupts before each critical timing loop and reenable them after the delay is complete. If you use lots of delays, this may completely block the interrupt service capability of the computer. You will have to evaluate your own needs to find the best compromise.

Finally, software timing loops are not extremely accurate. They are good for maybe ±1% accuracy, but beyond that you really should go to a hardware real time clock. It is particularly difficult to get good results with short internal delays. Your delay can only be adjusted to ± one clock cycle and this becomes significant in short program sections.

Program timing is an important operation in some applications. Proper analysis demands a good understanding of both the hardware and software being used. In this article we have covered some of the basics of both timing program sections and designing program sections to provide fixed time delays. With a little practice you should be able to use these principals as you design and use your programs. ■

Letters to the Editor

from page 17

DEFINING BAUD . . . AGAIN

Be careful of your definition of *baud* and *kilobaud* in the glossary. Bauds and bits/second are not necessarily equivalent. You can have a communications circuit operating at 2400 baud, but the bit rate could be 1 to 10 times the baud rate depending upon the type of modulation equipment being used. The equivalency of 1 baud equals 1 bit/second is applicable for simple communication links only. The term baud which had its origin in the days of dc telegraphy is an archaic term while the unit bits/second is more precise and meaningful.

Harold Corbin
Rockville MD

73 MADE A MISTAKE?

Several months ago I sent in subscription requests for both 73 and *Kilobaud* (nee Kilobyte). I have received the first issue of *Kilobaud* and all congratulations are well deserved. It may well become the standard against which all similar magazines are measured.

As for my subscription to 73, I have not had any indication that you have received same; therefore, the attached subscription request is being resubmitted.

James Schwartz

P.S. Please do not forget the neophyte (me) throughout the long life your publications will obviously enjoy.

OPTOISOLATOR'S

Issue #1 of *Kilobaud* came this morning. I am very pleased with it as I am a Novice who previously thought *Byte* and *Ram* were forms of assault. Control applications are of special interest to me so Chris Bowick's article was the first I turned to. It is well done and the examples are well chosen, but in Fig. 5 the ground connection of the 74145 BCD to decimal is labeled 7 instead of 8 (picky, picky, picky)! Full wave output should be possible with 2 optoisolators for channel. Please comment on other direct connected output devices with higher power ratings. Keep up the good work!

Dick Williams
Los Angeles CA

continued on page 108

Bob Haas
2288 Blackburn Street
Eugene OR 97405

KIM-1 Memory Expansion

. . . adding memory to this popular system is a snap

When you get ready to sit down and write that construction article you've been putting off, let me suggest you go back and re-read this article on KIM-1 memory expansion as a good example of how to do it right. — John.

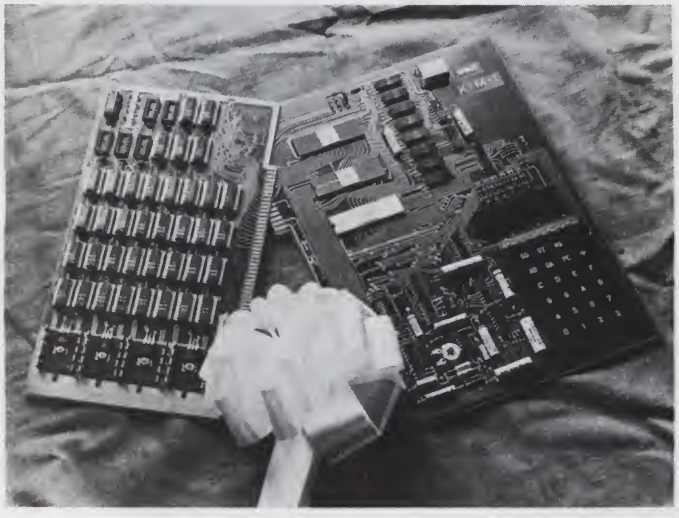
If you own an MOS Technology KIM-1 microcomputer and wish to add to its memory capacity, your first choices would be the KIM-2 (4K) and KIM-3 (8K) preassembled memory boards from MOS Technology. But, it is possible to modify Altair-compatible memory boards to work with KIM-1. One such board, the S. D. Sales 4K Low Power Ram Board Kit (available from S. D. Sales Co., P.O. Box 28810, Dallas, Texas 75228) is easy to modify. It can be connected to KIM-1 with the addition of only wire, connectors, four resistors, and a power supply. Therefore, it is an easy first step for the KIM-1 owner who is just beginning to expand his system. Furthermore this particular board can be modified without making any permanent changes to the board (i.e., no printed circuit trace cutting is required). The board can be restored to its original condition as an Altair-compatible board and resold to an Altair owner at a later time, if that should become desirable.

In addition to being easy to modify, the S. D. Sales memory board is an excellent value, being the most inexpensive kit of its kind that I am aware of. All parts are of high quality, including the solder-masked printed circuit board with plated through holes (like the KIM-1 board), and 21L02 high-speed (500 ns; within KIM-1 requirements) low-power RAMs. Address and data lines are fully buffered.

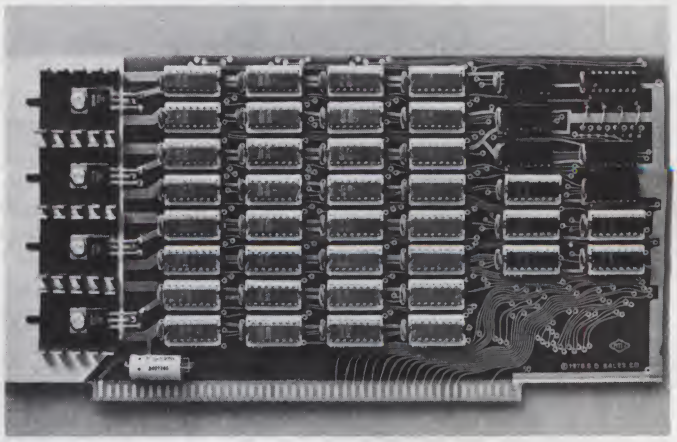
The board requires one ampere from an 8 volt (or so) unregulated power source. The power supply circuit shown in the KIM-1 User Manual *cannot* power both the memory board and KIM-1.

Theory of Operation

This modification uses the memory decoding already provided on the KIM-1 board. The added memory is logically concatenated with the 1K RAM on KIM-1; the result is 5K (5,120) bytes of contiguous RAM, with addresses from 0000 to 13FF (hex).



KIM-1 and the S. D. Sales 4K Low-power Memory Board pose for their formal wedding portrait.



The memory board after assembly and modification. Note that in the column of IC sockets on the far right two are empty, and that in the next column, three are empty. These ICs are omitted as part of the modification. Note also the address jumpers near the upper right corner of the board.

A simplified schematic of the modified memory is shown in Fig. 1. Most of the 21L02 RAMs have been deleted from the diagram of the memory to clarify it and make the basic organization easier to see. Most of the decoding circuitry on the original board is deleted as part of the modification. The only remaining portion of that circuitry, IC34, is shown. (In this discussion and in the modification instructions the IC numbers are those used in the kit manufacturer's documentation.)

The thirty-two 21L02 RAMs are connected in four rows of eight RAMs each. Each row stores 1,024 (1K) bytes, the eight RAMs in each row composing the eight bits of a byte. Only the left-most two columns and lower two rows are shown in the diagram. As can be seen, the four RAMs in each column have their inputs and outputs tied to common lines, and the eight RAMs in each row have their chip enable lines tied to common lines, which are the K1, K2, K3, and K4 lines from the memory decoder on KIM-1. When no access is being made to the added memory, K1-4 inputs are at a high (logic 1) level, pulled up by added resistors R1-4.

When the decoder on the KIM-1 detects that a reference is being made to an address in the range of 0400 to 13FF (hex), it *pulls down* (to a logic 0) one of the K lines. The eight 21L02s whose chip enables are connected to that particular line are enabled for reading or writing (which one is not known at this point). Eight 21L02s are enabled at once to compose the eight bits of a byte.

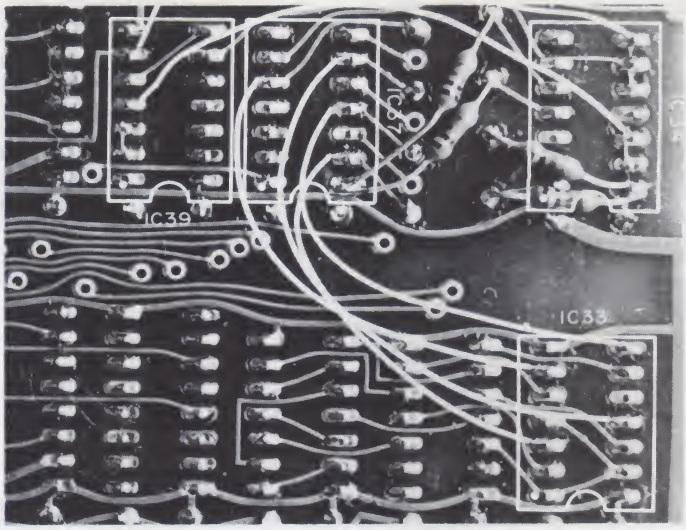
The K lines are also connected to a NAND gate, IC34a. It is shown as an OR gate with inverted inputs rather than as a NAND gate because it is performing an OR function: to detect if K1 *or* K2 *or* K3 *or* K4 goes low. When one of them does, IC34a's output will go high, denoting that one of the four rows of 21L02s has been enabled. IC34a's output can be called *board select* and it is labeled as such on the diagram. IC34b's inputs are R/W (read/write) from KIM-1 and board select. (IC34b is not used on the original board, but is used here because of the modification jumpers installed. IC34b is a four-input gate, but it is used as a two-input gate by connecting its inputs in pairs.)

The R/W line is high during processor read cycles and low during write cycles. IF R/W is high, denoting a *read*, and if board select is high, denoting an access to the board, then the output of IC34b will be low. This will enable Tri-state buffers IC38 and IC43, and the outputs of the enabled 21L02s will be connected to the system data bus, from which the processor will read the data.

During a write to the board, one of the K lines will be low and board select will be high. However, R/W will be low, therefore the output of IC34b will remain high, IC38 and IC43 will not be enabled, and the 21L02 outputs will not be connected to the data bus. This is as it should be, since the processor is supplying data to the data bus during a write cycle. The RAM-R/W line from KIM-1 is connected to the write inputs of the 21L02s; it goes low during the time that data is valid for a write cycle. Even though RAM-R/W is connected to all thirty-two 21L02s, data is written into only the eight that have their chip enable inputs low.

KIM-1 address lines AB0 through AB9 are applied to the 21L02 address lines by sections of IC40 and IC42, which are continuously enabled by their enable inputs being grounded. AB0 through AB9 are decoded internally by the 21L02s to select one of the 1,024 bits within each 21L02.

If you look at the KIM-1 schematic, you will see that the inputs and outputs of the on-board memory 6102s (which are 21L02 equivalents) are *split* by Tri-state buffers U13 and U14 (74125s) just as the 21L02 inputs and outputs are *split* by Tri-state buffers (8T97s) IC38 and IC43 on this memory board.



An enlargement of the portion of the back of the board in which the modification jumpers are installed. The four added resistors, the six insulated-wire jumpers, and the two bare-wire jumpers at IC34 and IC39, are visible.

Modifying and Connecting the Board

Begin the modification by constructing the board according to the instructions supplied. I repeat here the warning that appears in the instructions: "Assembly of this board requires experience in the soldering of very fine connections." A microspade soldering tip, such as the UNGAR PL-114 (available from James Electronics), is a great help. The solder mask on the board helps prevent solder bridges, but care must be used in the construction. Mount and solder the regulators to the board, but do not insert any ICs in sockets yet. All jumpers are installed on the back of the board (see photo). From this side of the board the view is of row upon row of IC socket pins, and it is difficult discerning which row belongs to which IC, so use care in installing the jumpers. Modification proceeds as follows:

1. Using a short piece of small diameter bare wire (such as #30 wire-wrap wire, stripped) solder a jumper between IC34 pins 6, 9, and 10. Solder a similar jumper between IC39 pins 2 and 3.

2. Using insulated wire, solder a jumper between IC34 pins 12 and 13 and IC39 pin 4. Solder a jumper between IC34 pin 8 and IC39 pin 6.

3. Solder four insulated wire jumpers between the following pins of ICs 37 and 33: IC37 pins 13, 11, 9 and 5 to IC33 pins 3, 8, 11 and 6, respectively.

4. Starting from the back of the board, insert one lead of each of four 560 Ohm, ¼ Watt resistors through the holes marked a, b, c, d nearest IC34. On the socket side of the board, bend the end of the resistor lead that comes through hole a near IC34 and pass it through hole a near IC37. Clip off the excess lead and solder both holes. Repeat this process for holes b through d. The other lead of each resistor must be connected to +5 volts. This is best done by bending the bodies of two of the resistors toward the connector edge of the board, clipping off the excess lead, and soldering them to IC37 pin 14. The other two resistors can be bent away from the con-

nector edge and soldered to IC34 pin 14.

5. Insert the 21L02s, IC34 (a 74S20), and ICs 38, 40, 41, 42 and 43 (8T97s). ICs 33, 35, 36, 37 and 39 are not used and must be omitted.

KIM-1 Expansion Connector	to	Memory Board Connector
pin A (AB0)		pin 79
pin B (AB1)		pin 80
pin C (AB2)		pin 81
pin D (AB3)		pin 31
pin E (AB4)		pin 30
pin F (AB5)		pin 29
pin H (AB6)		pin 82
pin J (AB7)		pin 83
pin K (AB8)		pin 84
pin L (AB9)		pin 34
pin Z (RAM-R/W)		pin 68
pin V (R/W)		pin 47
pin 8 (DB7)		pins 43 *and* 90
pin 9 (DB6)		pins 40 *and* 93
pin 10 (DB5)		pins 39 *and* 92
pin 11 (DB4)		pins 38 *and* 91
pin 12 (DB3)		pins 42 *and* 89
pin 13 (DB2)		pins 41 *and* 88
pin 14 (DB1)		pins 35 *and* 94
pin 15 (DB0)		pins 36 *and* 95
KIM-1 Application Connector		
pin C (K1)		pin 33
pin D (K2)		pin 85
pin E (K3)		pin 86
pin F (K4)		pin 32

Connection to the memory board requires an Altair bus 100-pin connector, available from several mail-order sources or computer stores. Connection to the KIM-1 expansion port requires a 44-pin connector, available at Radio Shack, part number 276-1551. Connections are as follows:

The above connections can be made with small diameter wire and should be no more than a foot long to minimize noise problems. The ground on the memory board, pins 50 and 100, should be connected to the ground on KIM-1 at expansion connector pin 22 or application connector pin 1 with number 18 or heavier wire. The negative side of the memory power supply, a source of about 8 volts unregulated at 1 ampere, should be connected to memory board pins 50 and 100. The positive side of the supply should be connected to memory board pins 1 and 51. These connections should be made with number 18 or heavier wire.

Note: the jumper from KIM-1 application connector pin K (decode enable) to pin 1 (ground) that is installed as part of the start-up instructions in the KIM-1 User Manual *must* be left in place.

Testing

Plug KIM-1 into the application and expansion connectors but do not plug the memory board into its connector. Apply power to KIM-1 and verify that all on-board functions work, e.g., try writing and reading cassette tape and altering and displaying memory. If any KIM-1 functions fail, remove power, unplug the expansion connector, and try again. If the failure disappears, the problem is probably a short between two pins on the expansion connector. If everything seems to be functioning, remove power, plug the memory board into its connector, and apply power to both the memory and KIM-1. Again verify that all on-board functions of KIM-1 operate properly.

Next, verify that the new memory can be examined and changed from the KIM-1 keyboard — enter AD (address) 0, 4, 0, 0 then DA (data) 0, 0, 1, 1, 2, 2, 3, 3, . . . ,E, E, F, F to verify that all bits are usable at address 0400. Repeat for addresses 0800, 0C00, and 1000.

If all addresses tested fail, suspect a problem with the data bus (DB0-DB7) wiring, or the RAM-R/W or R/W wiring. If some addresses work but others fail look for a problem in one or more of the K lines.

If these tests are successful, further testing must be done with a program. The logic of that program should be:

1. Store hex 00 in all locations to be tested. Verify that each location accepts the 00.
2. Store hex FF in the first location to be tested, verify that the FF has been accepted, then test all other locations to see that they still contain 00. Clear the first location to 00 store FF in the second location and test all other locations (including the first) to see that they still contain 00. Repeat for all remaining locations to be tested.

Testing by this kind of program will reveal problems in the address wiring (AB0-AB9, and K1-K4), or defective RAMs.

Conclusion

When I ordered my memory board from S. D. Sales, I did not know how difficult or easy it would be to adapt it to my KIM-1 system. I was pleasantly surprised at how easy it was, and I suspect that many other Altair-compatible accessories can be just as easily attached to KIM-1. With a bit of ingenuity, we KIM-1 owners can have the best of both worlds. ■

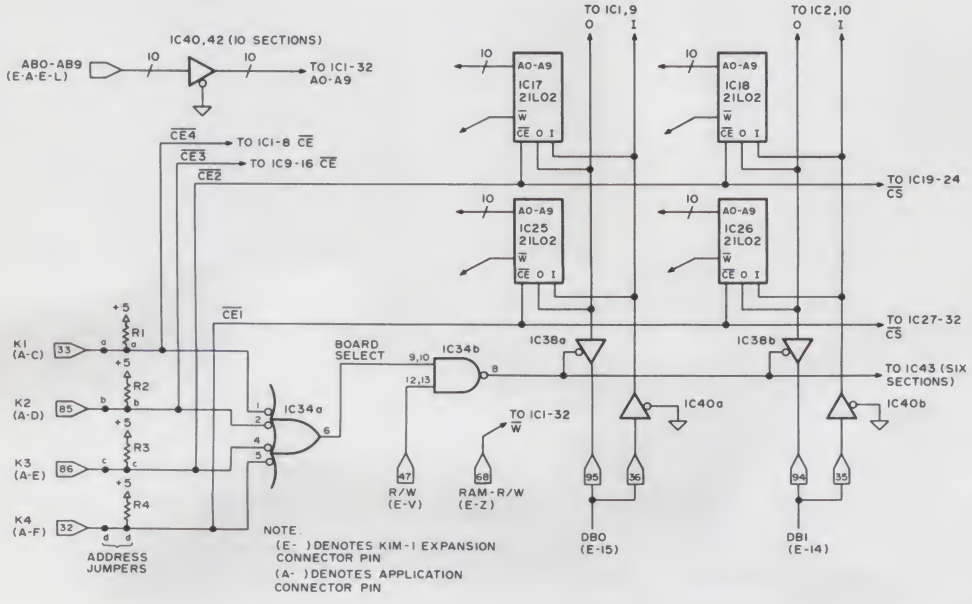
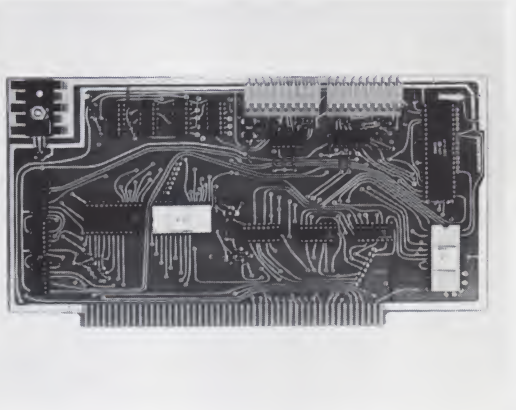


Fig. 1. Simplified schematic of modified memory board.

DIGITAL DATA RECORDERS

USING 3M DATA CARTRIDGES



MODEL 3M3 – $199.95
(Price increases to $220 effective 1 April 1977)

2SIO (R) CONTROLLER – $190.00

MODEL 3M3
Featuring the radically new "Uniboard" method of construction for data cartridge drives. The major computer makers are changing to cartridges at a rapid pace because of the freedom from binding and greater data reliability. Operates in the phase encoded self-clocking mode which provides greatly enhanced freedom from speed variation problems and allows 100% tape interchangeability between units.

Uses the 3M Data Cartridge, model DC 300. This cartridge contains 300 feet of .250 tape in a sealed plastic container. Using four tracks you can record nearly 2 megabytes of data on a cartridge.

SPECIFICATIONS:
Full software control of record, play, fast forward and rewind. LED indicates inter-record gaps. EOT and BOT are sensed and automatically shut down recorder. Feedback signals send reset and inter-record gap signals back to the computer so that software searching for inter-record gaps at high speed can be accomplished. Can also be operated manually by means of the switches on top which parallel the software control signals. $199.95 until April 1, 1977. $220.00 after April 1, 1977. Includes Phase Encoder Board (ACI).

FOR 8080, 8085, AND Z80 USERS
Comes complete with software listing for the programs in the 2SIO (R) ROMs. Can be controlled by any of the commonly used I/O boards. Send for complete documentation and interfacing instructions on 3M3 and 2SIO (R) ($3.00). These programs provide full software control.

2SIO (R) CONTROLLER (BOOTSTRAP ELIMINATOR)
This is a complete 8080, 8085, or Z80 system controller. It provides the terminal I/O (RS232, 20 mA, or TTL) and the data cartridge I/O, plus the motor controlling parallel I/O latches. One kilobyte of on board ROM provides turn on and go control of your Altair or Imsai. No more bootstrapping. Loads and Dumps memory in hex on the terminal, formats tape cartridge files, has word processing and paper tape routines. Best of all, it has the search routines to locate files and records by means of six, five, and four letter strings. Just type in the file name and the recorder and software do the rest. Can be used in the BiSync (IBM), BiPhase (Phase Encoded) or NRZ modes with suitable recorders and interfaces. $190, wired and tested; $160, kit form.

AUDIO CASSETTE INTERFACE (ACI)
This is the phase encoding board used in the 3M3. Additional components on the board enable you to use audio recorders in the KC standard or the new PE 2400 (2400 baud) systems. Can also be used for Tarbell if you have an 8251 Intel I/O chip. Required if you use an audio cassette with the 2SIO (R) above. $50, wired and tested; $35, kit form.

For 6800 Users: Software programs and I/O board for SWTP are under development. Limited software available now. Ask for 6800 data with $3.00 Documentation package. These programs will provide full software control.

CARTRIDGE AVAILABILITY
Cartridges are made by 3M, ITC, Wabash and others. They are available at all computer supply houses and most major computer service centers. We can supply them at normal current list prices.

"COMPUTER AID" and "UNIBOARD" are trademarks of the NATIONAL MULTIPLEX CORPORATION. The 3M Data Cartridges are covered by 3M Patents and Marks. "UNIBOARD" Patents Pending.

OVERSEAS: EXPORT VERSION – 220 V – 50 Hz. Write factory or – Megatron, 8011 Putzbrunn, Munchen, Germany; Nippon Automation 5-16-7 Shiba, Minato-Ku, Tokyo; Hobby Data, FACK 20012, Malmo, Sweden; G. Ashbee, 172 Ifield Road, London SW 10-9AG.

For U.P.S. delivery, add $3.00 Overseas and air shipments charges collect. N.J. Residents add 5% Sales Tax. WRITE or CALL for further information. Phone Orders on Master Charge and BankAmericard accepted.

Canadian Distributor:
Trintronics Limited
186 Queen Street West
Toronto, Canada M5V 1Z1
Tel: (416) 598-0262

BANKAMERICARD master charge

NATIONAL MULTIPLEX CORPORATION N-1

3474 Rand Avenue, South Plainfield NJ 07080, Box 288. Phone (201) 561-3600 TWX 710-997-9530.

William Cattey
39 Pequot Rd.
Wallingford CT 06492

Heavy Duty Power Supply

. . . juice for an entire system

William has come up with a design for an inexpensive Altair bus power supply (the "Hobby 1") which should do nicely for adding an Altair bus interface to another computer, beefing up an Altair bus machine, expanding a full-blown system, or perhaps for a piece of Altair bus test equipment. Seems as though the formulas and design considerations he mentions in the article would make for a nice program to have around the lab for the next time you need to build a supply (the video display would ask you for all the parameters . . . and then after calculating everything you would get printed a hard-copy list of all the parts need). — John.

How much must a good hobbyist power supply have to cost? Right now, vendors say a minimum of fifty dollars for five volts at five Amps. The Hobby 1 is based on information supplied by Steve Ciarcia over the phone and in the April, 1975, issue of *The Micro-8 Newsletter*. The unit uses a high current transformer that has all of the right voltages. The transformer is available from Delta Electronics for $8.95. I am told that they have five thousand of them in stock at this time. The two bridge rectifiers required are available for about $2.00 each and the necessary filter capacitors are available for about $4.00 each, bringing the total cost to well under twenty-five dollars for the complete unit.

How much power supply do you get for twenty-five dollars? The Hobby 1 supplies the three voltages needed to support an Altair 8800, Imsai 8080, or any other Altair-bus computer. There is a +8 volt supply rated at five Amps, and a ±16 volt supply that will support up to 2.5 Amps between the two voltages. The Altair 8800a supports +8 at 8 Amps and ±16 at one Amp between the two voltages. Therefore, although the Hobby 1 is not as hefty at the +8 level, there is twice as much power available at the other two voltages. At twenty-five dollars each, you can afford to build a pair! In order to support as much as an Altair 8800b is capable of (+8 at 18 Amps and ±16 at two Amps each) four Hobby 1 units would be needed. The total cost of the four ($100.00) is significantly lower than the kit Mits sells to expand the 8800a supply to the 8800b rating ($147.00)!

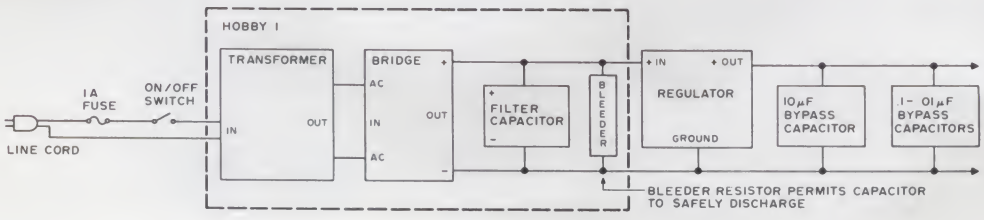


Fig. 1. Power Supply block diagram.

Theory of Design

Although it is not essential to the construction and operation of the Hobby 1 power supply, the theory and math behind the design are covered here. In this way, you can design custom power supplies to fit special voltage and current needs or to see if you can substitute parts that you have on hand for those in the parts list. For those who do not care about the theory, I have included the parts list and information on construction and installation.

What do we need for a complete power supply? Fig. 1 shows a block diagram that includes all of the necessary components. The Hobby 1 contains everything within the dotted box. Circuit boards designed for Altair bus compatibility contain everything to the right of the box. The user must come up with everything to the left. As shown in the diagram, a power supply needs a transformer to change the input ac

to something slightly above the desired level. A slightly higher level is necessary because it is much easier to drop voltage and current than to raise it. The ac output of the transformer is converted to pulsating dc by the rectifier. The full wave bridge rectifier is the most efficient because it utilizes the whole waveform and input voltage to generate the output voltage. The pulsating dc goes from some peak level to zero. The filter capacitor delays the drop to zero long enough for the capacitor to be recharged. Depending on the value of the capacitor, the output will always remain some voltage above zero. The range between this voltage and the peak is called ripple. It is the job of the filter capacitor to keep the output at a level where the regulator is effective. Most regulators require an input at least three volts above the desired output level. The two capacitors after the regulators are called bypass capacitors. They ensure that no unwanted spikes of ac are permitted to damage any circuitry. The 10 uF capacitor is installed as close to the regulators as possible. There should be (according to Don Lancaster's *TTL Cookbook*) one .1 uF to .01 uF capacitor for every four small scale packages, every two medium scale packages, or one for every large scale package.

As mentioned earlier, the design is based on information and equations supplied by Steve Ciarcia. Given that we have a Delta Electronics K9906 transformer, and that we want +8 volts and ±16 volts outputs, it is possible to use some math to determine the values of the other components. Fig. 2 shows the complete schematic of the Hobby 1 power supply. It contains a single +8 supply and a dual 16 supply. Although there are other ways of generating the ±16 supply by using other circuitry, the method I have used is simpler than making two completely separate supplies.

The first things that must be determined are the maximum ratings for bridge rectifiers. For maximum ease, all maximum ratings will be based on ratings found for the 18 volt transformer winding. In this way, if the transformer somehow gets connected to the other set of components, they will not be blown out. The maximum current across the bridge will be about 250 Amps which is the peak voltage (about 25 volts) divided by the resistance of the transformer winding (.1 Ohms typical). In order to protect against something like the transformer shorting out and applying the 115 volt ac input to the rectifiers, I have selected those with a 200 volt rating. This is typical of the types of safeguards built into all commercial power supplies. The MDA 980-3 is rated for 200 volts and 300 Amps (peak) which is just what is needed. Table 1 shows how the filter caps were rated for voltage based on the absolute peak input voltage to them which might, if ever the bridge rectifiers shorted out, be the peak output voltage of the transformer.

To determine the necessary value of the filter capacitors, we use the three equations in Table 2. The capacitance, as stated earlier, must keep the output above 8 volts for the +8 supply and above 16 on the ±16 supply (3 volts above the desired regulator output). Therefore, we must know the peak output voltage of the rectifier and subtract from it the desired voltage output. This value is plugged into the capacitance equation as ripple. The peak output of the rectifier is the RMS value of the transformer minus the drop across the diodes in the bridge (typically 1.5 volts) multiplied by 1.4. The capacitor's value *is a function of the* charging time (a constant .0083 for a bridge rectifier), the ripple, and the current to be drawn. Table 2 also shows the values substituted into the equations for both of the supplies.

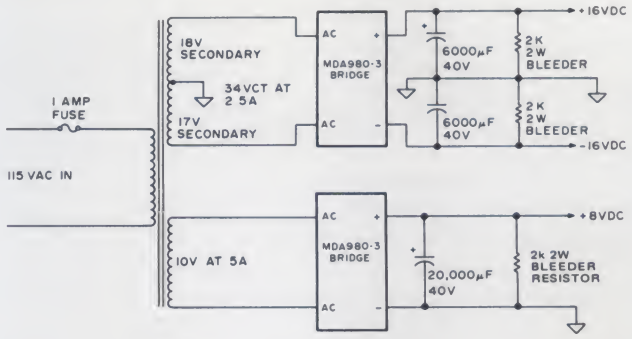


Fig. 2. Hobby 1 schematic diagram.

For the absolute maximum voltage, we take the peak output of the biggest secondary and add at least 25%. In computing the max RMS the maximum line voltage of 125 volts is used (the variation is usually from 105 V ac to 125 V ac).

MAX RMS = (Vout x max line voltage) ÷ rated input of primary
= (18 x 125) ÷ 115
= 19

Peak = RMS x 1.4
= 19 x 1.4
= 26.6

Max = Peak + 25% of Peak
= 31.24 or approximately 40 volts

Table 1. Formulas for determining voltage rating of filter capacitors.

equ. 1 — Peak = (Eout - 1.5) x 1.4
equ. 2 — Ripple = Peak - Out
equ. 3 — C (infarads) = (Current Rating x .0083) ÷ Ripple

Out = + 8 VDC	Out = + 16	Out = - 16
Eout = 10 VRMS	Eout = 17 VRMS	Eout = 17 VRMS
Current = 6 Amps	Current = 2.5 Amps	Current = 2.5 Amps
Peak = (10 - 1.5) x 1.4	Peak = (16 - 1.5) x 1.4	Peak = (16 - 1.5) x 1.4
= 11.9	= 21.7	=21.7
Ripple = 11.9 - 8 = 3.90	Ripple = 21.7 - 16 = 5.7	Ripple = 21.7 - 16 = 5.7
≈3	≈5	≈5
C = (6 x .0083) ÷ 3 = .0166 Farads	C = (3 x .0083) ÷ 5	C = (3 x .0083) ÷ 5
= 16,600 ufd	= 4,980 ufd	= 4,980
add 10%	add 10%	add 10%
= 18,260 (minimum)	= 5,478 (minimum)	= 5,478 (minimum)
≈20,000 ufd	≈6,000 ufd	≈6,000 ufd

Table 2. Formulas for determining the filter capacitor values.

The value of the bleeder resistors should be around 2k Ohm. Their purpose is to discharge the capacitors after the power supply has been shut off. In this way they do not discharge through you when you service or test the power supply! (Note: It is a good practice to wait three or four minutes after powering down to allow for total discharge of filters before removing and inserting boards.)

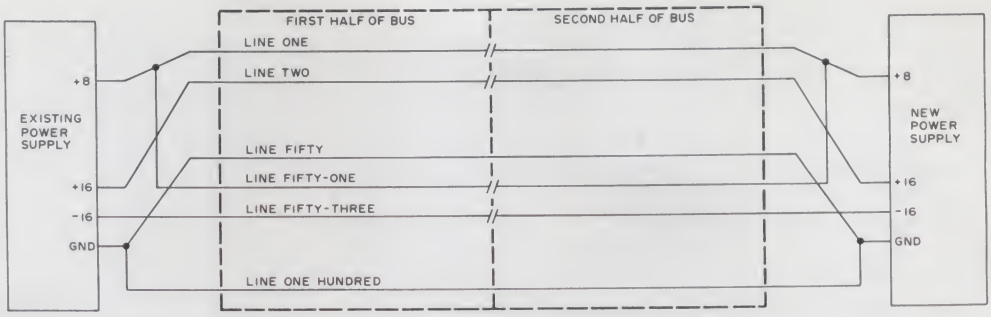


Fig. 3. Diagram illustrating dual supply connections.

Construction

The best way to put the thing together is to get a piece of phenolic board or perfboard, lay out the position of the components, and point to point wire them. Although you can waste your money on getting a circuit board etched, the supply is not really that critical. It would be a good idea to bring all of the connections out to a terminal strip though. Wire up the supply and power it up. Make sure that the outputs are at least as high as their rated voltage.

To connect the supply to an Altair bus, connect the +8 output to pins 1 and 51, the +16 output to pin 2, the -16 output to pin 52, and the ground line to pins 50 and 100. If you are using the Hobby 1 as a booster supply for your computer, you must isolate all of the outputs *except* the ground from the other power supply, because if they are not matched to each other exactly, one will *draw* current from the other. To isolate the power supplies, determine where on the bus the loads will match the ratings. Connect the Hobby 1 to the other end of the bus and cut the lines between the supplies at the point you determined would match the loads. A schematic representation of such an addition is shown in Fig. 3. The same arrangement can be used to build three or four onto the same bus. It is important to remember that only the +8, +16, and -16 lines are to be cut. Good luck and have fun! ■

Quantity	Description	Price
1	Delta #K9906 High Current Transformer	$8.95 ea.
2	Motorola MDA 980-3 Full wave bridge	2.00 ea.
2	6,000 uF electrolytic capacitors 40 volt	4.00 ea.
1	20,000 uF electrolytic capacitor 40 volt	4.00 ea.
3	2k Ohm 2 Watt resistors	.50 ea.
	Approximate total	$21.45

Table 3. Power Supply parts and price list.

COMPUTER COMPONENTS

5848 Sepulveda Blvd., Van Nuys, CA 91411 (213)-786-7411

Computer Components is dedicated to serving the needs of the computing community. We offer a complete line of computers and computer related products. We will ship anywhere in the world. Satisfaction guaranteed.

Hours:

Tuesday-Friday	10AM to 9PM
Saturday, Sunday	10AM to 6PM
CLOSED MONDAY	

Terms:
Call or write for discounts. We pay shipping on UPS orders over $25. All others add $2 for postage and handling.

We are authorized distributors for the following companies:
IMSAI • INTELLIGENT SYSTEMS • POLYMORPHIC SYSTEMS • CROMEMCO • TARBELL • KIM1 • TECHNICAL DESIGN LABS • SOLID STATE MUSIC • VECTOR GRAPHICS • RO-CHE SYSTEMS • CONTINENTAL SPECIALTIES • VECTOR ELECTRONICS • SAE • TEXAS INSTRUMENTS • SIGNETICS • NATIONAL SEMICONDUCTOR • ADVANCED MICRO DEVICES • SAMS BOOKS • HAYDEN • WILEY
...and more on the way. Always ask.

Digital Audio

... the next

Tom Scott
Uncalledfor Productions
700 Bay Road
Mill Valley CA 94941

Photo by Kadi Kiis.

"There's got to be a way to hook this up . . ."

I've noticed that there have been a lot of articles lately linking ham radio and home computers, and that's fine. I'm a ham myself. But how about home computers and the *other* great electronic hobby, hi-fi? You haven't seen any yet yourself? I thought not. Well, in the "Am I out here all alone?"

revolution in sound?

vein, I'd like to stir up some interest in bringing together the stereo-fiend that lurks in us all and the hobby-computer nut that we all seem to have become. Now, if it turns out that I *am* out here all alone, I'll go back into the woodwork. But if, as I suspect, there are hundreds of budding Audio-Computophiles waiting to make contact, then let's communicate. We've got a lot to tell each other and even more to ask. And to get the ball rolling, I'd like to offer some basic thoughts on digital audio. Now, I'll tell you out front that there is not much in the way of cards or kits yet, there are no programs that I know of, and there aren't even many schematics to work from. But the possibilities are tantalizing!

Digital Audio

"Digital audio?" you say, "sounds paradoxical to me — like comparing apples and pineapples. Why would anyone want to mix up music and digits? Why take the smoothly-continuous sound waveform and machine-gun it with 1s and 0s until the musical fabric is chopped to bits?" That's how I felt a couple of years back when I was given a peek at a system that purported to take the full fidelity of music and convert it to a stream of numbers, storing every nuance, inviolate for all time. That's what they said, anyway. It turned out that the inventors of the system were (only) a few months ahead of their time. They went out of business. Less than a year later, however, another firm demonstrated a professional audio delay that used the same digital-audio principles. Since then the field has been growing steadily, and in the process it has made me a believer.

"What's so great about having numbers describe sound?", is your next question.

"My stereo sounds pretty good . . ."

There are a number of reasons why digital audio is fast approaching but the basic one, at least to my way of thinking, is the inelegance of the present system of tape recording and vinyl record pressing; it's just so mechanical. What do I mean by that? Imagine that you are an investor. An inventor arrives at your door with a system for music storage and reproduction that he describes like this:

"Ahem . . ."

A System of Sound Recording

"First, we get some plastic and make a long thin ribbon, mirror polished and very precise. Then we grind up some iron and stick it to the plastic and wind the whole thing up on a roll. To handle this we get some precision motors and bearings and an exactly milled frame to hold them together with flywheels, belts, rubber rollers, solenoids, switches and levers. Now here comes the good part: we wind some coils, you see, and make electromagnets that touch the ribbon as it goes by. Special amplifiers will take sound waveforms from the microphone and energize the electromagnets to magnetize the tiny bits of iron as they are spooled past."

At this point he mentions a supersonic whistle to really confuse things and you begin to wonder if you have a nut on your hands. But he isn't done yet:

"Next, another set of coils generates a tiny voltage as the little magnetized particles move past, and this in turn is amplified to yield sound!"

You don't want to invest? Sounds a bit Rube Goldberg, you say? But that's the way a tape recorder works: Bing Crosby invested 25 years ago and the thing has been growing strong ever since. Tape recording has come a long way, to be sure: from simple one-track lo-fi machines to the 24 track, servo-controlled, noise-reduced, studio monsters of today — along with such spin-offs as the 8-track cartridge, the cassette, and the Dictaphone. Even the video tape recorder stems from the same technology. Kind of like the mechanical adding machine, just before the pocket calculator came along.

The lathe system that cuts record masters, the plating and stamping process, the vinyl stereophonic disc, and the stero playback cartridge are all equally nonobvious, extremely mechanical, precise, finicky, fine-tuned, complicated systems. They've carried us a long way from the Gramophone era to the present state of the art, but how much further we can

pursue this line of thinking? Sure, there will probably be a new plastic along shortly that will make quieter, cheaper discs. Tape manufacturers will surely come out with a new generation of tape with better life, less noise. But this sort of fix is like a Band-Aid on the leg of a draft-horse that should be put on the milk runs for his remaining good years. His time for pulling the full load might be drawing to a close.

An Alternative

Imagine instead that we convert the sound waveforms into numbers and store them away in a memory device. Then when we want a playback we simply withdraw the numbers from the data bank and reconvert them to sound waves; what could be more elegant? No moving parts. Conventional tape recordings suffer from tape hiss, limited dynamic range, tape speed variations like flutter and wow, frequency response errors, and phase shift. Digital audio could put these problems into the same category as dull cactus needles for your grandfather's Gramophone. Every successive copy made of a conventional tape recording loses quality. Digital audio programs could be copied dozens of times, and every copy would be indistinguishable from the original. They are just strings of numbers, remember, and we have ways to catch errors in copies of strings of numbers, right?

"Wow!" you say. "I want one . . . or maybe I already have one . . ." as you eye your Altair/disc system, currently entertaining the neighborhood kids with "Guess the Number" and "Hunt the Mangel-Wurzel."

"At last! A really worthy project."

Well, I must confess, now that I have your interest, that digital audio recording is not quite that easy. It's not here yet, but it might be right around the corner.

Around the Corner

Oops, did I say it's not here yet? Looks like I spoke a bit too late. There are actually several digital audio systems in existence and rumors of more to come any day. The Japanese have actually released several LPs of classical music recorded with digital techniques. The BBC have been working on a digital audio distribution system that will eventually cover the British Isles. In this country, Dr. Thomas Stockham demonstrated a digital recorder at the Audio Engineering Society Convention in New York in November '76.

Furthermore, he is working on the development of an all digital recording studio in Salt Lake City. So, there are people who have been able to get started with this. Of course, they all have bigger machines, more employees, bigger budgets, and more experience than we have, but when have true hobbyists ever let things like that stand in their way? The way I figure it, this whole hobby was an impossibility a couple of years ago. So let's look into a basic system.

How Does It Work?

Before we can record digitally we must convert the varying voltage that is the output of a microphone or preamp into a string of numbers. This is termed an analog- (continuously varying) to-digital conversion, usually shortened to A-to-D (or A/D) by the insider. There are several ways of performing this function, but for this first look, let's consider the A-to-D converter as a black box and concern ourselves only with the input and output (refer to Fig. 1). In order for the converter to work properly, the varying input is chopped into short time segments just as a repetitive strobe light stops the motion of a falling object. This chopping is performed by a *sample and hold* circuit; and, in order not to miss the smallest details of the sound waveform, the circuit must sample at a rate more rapid than twice the highest frequency present in the input analog signal. This rule is called the Nyquist Criterion, after the communications pioneer who developed the idea.

So, the sampling rate dictates the upper limit of the frequency response. A sampling rate of 50,000 samples per second could yield a frequency response equal to the best of the professional tape recorders. 20,000 samples per second yields a response like that of an ordinary cassette recorder. After we have chosen a sampling rate, we must insure that the sample-and-hold never sees frequencies higher than it can handle. Rather than ignore frequencies too high for it, the A-to-D process would generate errors. To avoid this, the analog input is conditioned by a sharp cutoff, low pass filter which precedes the sample-and-hold.

Each sample of the input waveform is passed on to the A-to-D converter where a number is generated that is proportional to the amplitude (volume, if you will) of each of the thousands of samples per second. These numbers are the stream of data which will be stored in the recording process. Sounds pretty easy so far, but there is a problem that immediately shows up if you try to use your 8080 microcomputer as the basis for a digital recorder. How big a number is necessary to fully describe all the possible

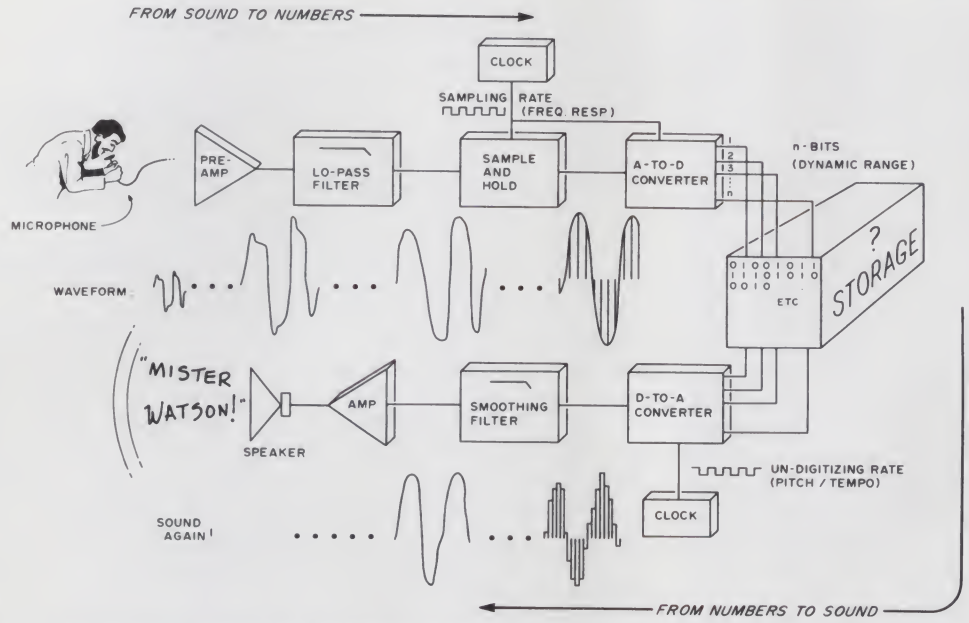


Fig. 1. From sound to numbers and back again. A simple sound storage and retrieval system . . . or maybe not so simple.

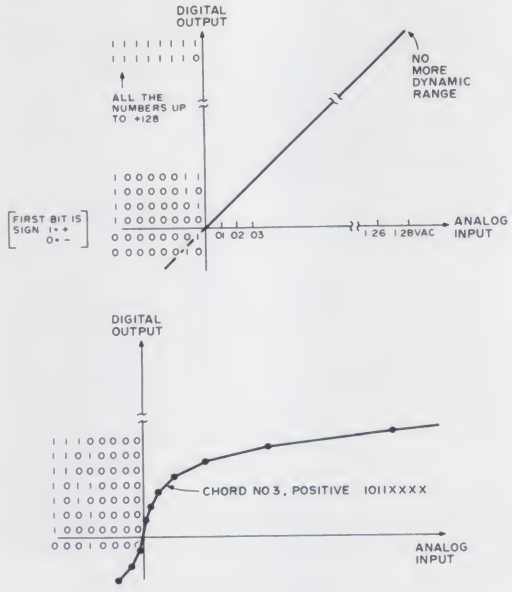
sample amplitudes?

My 8080 can easily store a number up to 8 bits long, which translates into 256 different possible sample values. If I decide to measure above and below zero volts, in 1/100 volt steps, I could characterize a signal between -1.28 and +1.28 volts.

"Not bad," you say. "My stereo power-amp puts out its maximum power with an input in that range . . ." Well that might be fine for the highest levels, I'd reply. The problem would show up at lower volume levels, where small changes couldn't be characterized by the relatively large steps of 1/100 volt. This is called *quantization error.* We must remember that our task is to store a stream of data that approximates the original input quantity so closely that the human ear won't hear any mistakes (distortion). We need to ask: What is the smallest voltage change that we'd be able to hear as a step in volume?

For this type of consideration, most experimenters use the decibel system of volume measurement. This logarithmic system of expression fits our sound perception better than the simpler linear voltage scale, because we readily perceive small volume changes at low volume levels; changes that would go unnoticed at high volume. The volume change produced by an increase in voltage from 1.00 volts to 1.01 volts would be imperceptible, while the same 0.01 volt change from 0.02 to 0.03 volts would be easily recognized.

"Of course," you say. "The first is only a 1% change while the second is 50%." Bravo! You're right with me. An audio person might use the dB system, and say the former is less than 1/10 dB while the latter is more than 4 dB; but both observations come down to the same thing. At this point you might like to get out a text and read up on the decibel. It's been covered many times,



(a) An example of linear encoding A-to-D "good for simplicity, repeatability, precision, and ease of data manipulation later."

(b) The hybrid, companding DAC-76 encoder A-to-D "for increased dynamic with economy of bits." First bit is sign, next 3 bits specify which chord, last 4 bits sub-divide each chord into 16 parts, linearly.

Note: At lower input levels the encoding steps are closer together, like human hearing. At lower levels we perceive smaller absolute (non-relative) changes.

Fig. 2. Analog-to-digital encoding.

but I'll recap some good points for us to keep in mind:

1. One dB is about the smallest change we can ordinarily perceive, while 3 dB corresponds to a doubling of the volume or sound pressure level.

2. If we assign the value 0 dB to the softest sound we can hear, then the loudest sound we can stand works out to about 130 dB. This 130 dB range of softest-to-loudest is called the *dynamic range.* The term can be applied to a piece of electronic gear as well as to human hearing: A good LP disc might have a dynamic range of 70 dB, a professional tape recorder about 80 dB. In these cases the *softest* is the smallest signal that can be heard above the inherent noise of the device, and the *loudest* is the highest volume it can produce without adding distortion.

3. The 130 dB change of the human hearing dynamic range might be modeled in a piece of electronic gear as the range from 10 u volts* to 30 volts. That's better than a million possible 10 u volt steps. To cover all these possibilities we'd need a number with around 20 bits! Looks like the human ear is still beyond an easy hardware simulation.

But what about a good tape recorder? We said that the dynamic range there is only about 80 dB. Let's say we use the range 1 mV** through 10 volts. In 1 millivolt steps that's only 10,000 possibilities. This we could do with about 14 bits. Still a lot of bits for an 8080 to process, but in fact the experimenters I mentioned earlier agree that to out-perform conventional audio systems will take 15 or 16 bits.

A good question for you to ask here is: "If perception works on a dB-style scale, and there are 130 barely perceptible steps to the human dynamic range, why not make our digital scale simply run in dB?" This is a very real possibility, and the only problem is practicality in the circuitry of 1977. Precise A-to-D conversion and subsequent error-free D-to-A reconversion require that voltage measurements be exacting and repeatable. A precise voltage proportional to the logarithmic dB scale is not easy to produce, and at this point there is no chip that will do the job on a hobbyist's budget. Linear voltage references are much easier to handle since all the steps are the same size (see Fig. 2a). A voltage source can be divided to get a linear scale with repeatable results. Thus, even though linear-scale digitizing seems wasteful in bits, for now the practicality and repeatability of that system will hold out over dB or logarithmic digitizing methods.

There is a way, however, that we can get more resolution with our limited number of bits. A technique used by audio professionals to increase the dynamic range of their limited Analog systems involves raising the level of the softer sounds before recording them so that they are loud compared to system noise. When this recorded sound is played back, the soft sounds are electronically pushed back down to their former level and any added tape hiss or system noise is pushed down at the same time. This makes the overall system quieter, and is the operating principle of a

*A microvolt (uvolt or uV) is one millionth of a volt.

**A millivolt (mvolt or mV) is one thousandth of a volt.

family of noise-reduction devices, called *companders* (a combination of compressors and expanders). The dynamic range is compressed before recording and expanded before playback. Now this is a computer magazine, so I'll leave the Dolby and DBX systems for you to explore in hi-fi periodicals. They've covered them many times each.

However, the idea of compressing our audio before we digitize and expanding after we've undigitized is an easy extension of the same idea. The only problem we encounter is the same one we found in our dB-scale consideration. We must have a precise reference for the compression curve and apply the opposite curve with equal precision for the expansion. But maybe someone's taken care of this for us. How about this:

At Last, a Chip!

This is the closest I'm going to get to actual hardware for hobbyists in this article: manufacturers haven't even imagined Audio-Computo-philes yet. There is one exception I know of, though, so make of this what you will. Precision Monolithics sells a very interesting 18-pin DIP package called the DAC-76. It is an *eight-bit* companding A-to-D converter that offers *12 bit* resolution through a very clever combination of the logarithmic and linear scales we touched on a few paragraphs ago (see Fig. 2b). It would be a bit beyond the scope of this article to explain how you use a D-to-A converter together with a comparator and a successive approximation register to do A-to-D conversion, though I'm sure PMI would be happy to tell you how to do it and sell you the chips as well. Their "eight bits will get you 12" secret, however, is actually quite straightforward. They approximate the dB curve with eight (8) straight line segments, or chords (recall your geometry), each with carefully measured end-point voltages. Three bits of the eight available bits specify which of the eight chords contains the value in question. The next four bits subdivide that particular chord *on a linear scale* into 16 equal parts. The eighth bit specifies the sign of the sample, positive or negative. They claim a 72 dB dynamic range, about that of a high quality cassette recorder.

Storage

Now that we've completed the encoding process, storage should be easy, right? Wrong. Sure, almost any kind of memory will do — as long as it's immense. A little simple arithmetic will show you that the amount of data necessary to successfully describe a three minute 45 rpm monophonic record is staggering. For 10 kHz bandwidth, we must sample 20,000 times each second. Let's say we use the aforementioned DAC-76 to convert to 8-bit data words. Oh-oh . . . we're using 20 kilobytes of storage every second! And there are 180 seconds in three minutes: 3.2 megabytes! And the fi isn't even all that hi.

Well, now you know the worst: The current Achilles' heel of digital recording is storage space. Lack of cheap, fast, mass storage is the main problem that is keeping a digital audio recorder out of your living room. But let's not give up after we've come this far. Disc systems are getting cheaper every day. Your floppy could be the first in the world to repeat "Mr. Watson, come here. I want you," or "What hath God wrought?" or some other equally momentous thing.

Photographic storage can achieve tremendous bit densities with easy duplication. Perhaps a computer hobbyist will be the first to demonstrate a cheap, laser-encoded, photographic audio recorder. Manufacturers are promising megabytes of bubble memory at hobbyist prices in the next few years, so let's get our A-to-D systems ready.

Wait a minute, though. We've forgotten the cheapest mass storage around: our old faithful tape recorder. Oh, I know I knocked the heck out of them back in my first paragraph, but there might be some room for a hybrid system. If we were to use mag-tape for storage of digital data, we might wind up with the best of both systems. Tape motion inconsistencies could be easily evened out by a small buffer memory in the output circuitry. Data would be read into the buffer at whatever varying rate the old mechanical clunker could manage. The same data would be read out of the buffer at a precisely clocked rate for constant pitch and speed. Your local TV station does something like this with their video tape machines; they call it time-base-correction.

A good deal of experimentation might be necessary to find a suitable serial code to lay onto the tape; the hobbyist cassette interface standards just aren't fast enough. But then, we don't have to record *audio* at all. Why not dispense with pre-emphasis, bias and the other complexities of conventional audio recording and record the *data* directly onto the tape. It's both possible and practical; the process is called saturation recording. If a professional tape machine manufacturer were to perfect a digital tape recorder for audio, my guess is that this would be the system that he would use, at least for now. But let's side step recording for a while. What else can we do with digital audio?

What Else?

We may not be able to store hours of music yet, but we can use our limited memory to provide useful short-term storage. I mentioned earlier that one of the first digital audio products on the market was the delay-line. This is a device that encodes, stores and then decodes audio to provide controllable time delays, usually less than one second long. Today, many pop records are made using digital delay devices to achieve peculiar audio effects. If the sound of an instrument or voice is delayed by a fraction of a second, and recombined with the original sound, a doubling effect is produced. This used to be done using two tape recorders or by recording two performances of the same musical passage. This double-tracking was cumbersome and expensive; a high quality digital delay device is still rather expensive (3-5k$), but the operation is much simpler. If the delay is very short, on the order of a few milliseconds, the familiar whoosh of the phasing or flanging sound, familiar to pop music enthusiasts, can be produced. The digital delay is occasionally used to augment the sound reinforcement system in large concert halls. The signals fed to loudspeakers close to the stage are delayed so that the sound produced reaches the listener at the same time as the sound from speakers at some distance from the stage.

For your home music system, a delay of less than a second can produce the ambiance of a giant auditorium. Pseudo-quad experiments could be a lot of fun: delay some of the stereo program material and feed it to another set of speakers behind you. Instant concert-hall! We might feed some of the delayed output back into its own input to create Grand Canyon-style echo. Some enterprising hobbyist might even write software to combine random delays and echos, synthesizing true reverberation. There must be many more applications waiting for experimentation by talented amateurs. There are probably even marketable products there, waiting to be discovered.

The Digital to Analog reconversion is the last stage of the process. It is com-

plicated by the fact that the D-to-A converter tries to create a replica of the original input by a succession of stair-step-like discrete jumps in voltage. These steps would sound like high frequency distortion on the output, so a suitable high frequency roll-off filter is used to round off the corners of the steps and give us back a smooth, continuous waveform.

Finally . . .

Well, I'm almost done for now. Did I arouse your interest a little? As I said before, there's a lot for us Audio-Computo-philes to talk about. We haven't mentioned pitch or tempo changing at all, yet. Or how about multiplexing several audio signals onto one channel? Or bandwidth compression for more economical use of FM frequencies on the ham bands? How about sonic burglar alarms, or maybe sonar? Or translation of high frequency bat sounds, or low frequency whale songs into the human audio spectrum, with your IMSAI? How about an automated hi-fi testing service for your local stereo store? And then there are control systems for your home music system, using your computer. The possibilities go on and on. I want to hear about it all, and maybe you do too, so let's get in touch.

Want More?

If you'd like a simple rerun of the A-to-D-to-A scenario, I'd recommend Ralph Hodge's article "Will Audio Go Digital" in *Popular Electronics* August, 1975.

For some hardware ideas for saturation recording, try *Byte*, January 1977, "Saturation Recording Is Not All That Hard" by David Allen. He also offers a good bibliography.

For a look at commercially available A/D devices, the catalogues of Precision Monolithics (1500 Space Drive, Santa Clara CA 95050) and Datel Systems (1020G Turnpike Street, Canton MA 02021) are good ones, but there are many others. ■

Jim Huffman
Hufco
P.O. Box 357
Provo UT 84601

HI-LO

. . . impress your friends when they visit

The following program arose out of the need to entertain my neighbors when they came down to my basement to see my computer set-up. I had seen Hi-Lo games in various forms such as those used with the HP-65 calculators, and so on, and I felt that a Hi-Lo game written in BASIC would be nice. It's rather easy to win, and the player is able to develop some skill in his guessing techniques. Basically, the program is self-explanatory; that is, the machine contains all the instructions for operations except telling the operator that he must press carriage return after making any entry. If you would like, at step 35, you might put in a PRINT statement, "Don't forget to hit carriage return after every entry." The program is written for MicroBASIC, and the random number generator generates a whole number. Therefore, step 40 might have to be changed or modified slightly depending on the type of random number generator your particular BASIC provides. My MicroBASIC program tended toward the low numbers — that's why I have the times three multiplier built in for the value of X, and by dividing my random number by 1,000, I would always get a number between zero and one hundred. Step number 45 is a security number to make sure the random number generated doesn't exceed 100. The best thing to do is implement the program and if you find your random numbers are generally running either high or low, multiply or divide by some arbitrary number. The main features of this program are (1) it uses whole number arithmetic and thus will run in various MicroBASICs. And, of course, since it is written in MicroBASIC, it will also run in BASIC with very little modification except that your random number might contain a fraction. If so, you will need to use the INT function to change it to an integer. It might be a little difficult to ask someone to try to guess any number between 0 and 100 if that number might be some fractional number (i.e., 45.61498). Fig. 1 is a flowchart description of how the program works. Program A is a typical run of the program:

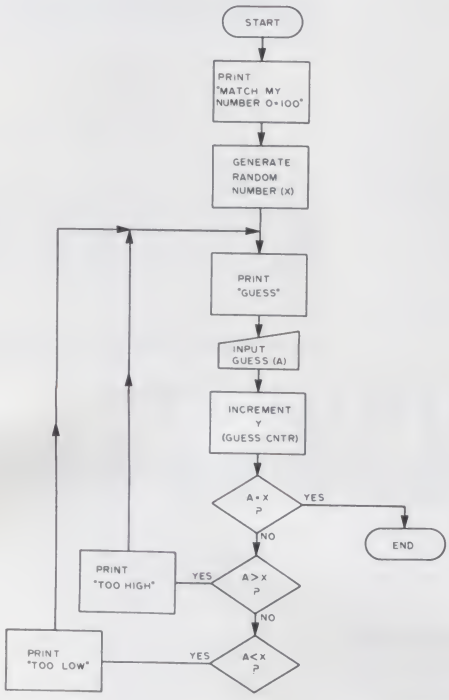


Fig. 1

```
#RUN
HI-LO
MATCH MY NUMBER (BETWEEN 0 and 100)
GUESS? 50
WRONG
TOO HIGH
GUESS? 25
WRONG
TOO LOW
GUESS? 37
CORRECT
37 IS CORRECT. IT TOOK ONLY 3 GUESSES.
DO YOU WANT TO PLAY AGAIN? 1 = YES, 0 = NO
?0
#READY
```

```
10    REM*** HI-LO
20    PRINT TAB (8); "HI LO"
30    PRINT "MATCH MY NUMBER (BETWEEN 0 AND 100)"
40    LET X = (RND/1000)*3
45    IF X >100 GO TO 40
50    LET Y = 0
60    PRINT
70    PRINT "GUESS";
80    INPUT A
90    LET Y = Y + 1
100   PRINT
110   IF A <> X GO TO 190
120   PRINT TAB (6); "CORRECT"
130   PRINT A, " IS CORRECT. IT TOOK ONLY";Y, "GUESSES"
140   PRINT
150   PRINT "DO YOU WANNA PLAY AGAIN? 1 = YES, 0 = NO"
160   INPUT B
170   IF B = 0 END
180   GOTO 30
190   PRINT TAB (8); "WRONG"
200   IF A > X PRINT TAB (6); "TOO HIGH"
210   IF A < X PRINT TAB (6); "TOO LOW"
220   GO TO 60
```

Program A

"It's the first book I've ever read about computers that I can understand . . ."

HOBBY COMPUTERS ARE HERE has continued to be the best selling of the books directed to the computerists. Your library is not *complete* without a copy . . . only $4.95 brings you a fantastic book for the beginner . . . whether that is you, your family, or your friends. It will help the beginner get into the world of microcomputers, a world of enormous fun. It is a complicated world and beginners need all of the really fundamental help they can get . . . like this book. Some chapters . . . What's a Computer?, History of Numbering Systems, Is Digital New?, How Computers Figure, What's That in Binary?, Computer Languages, How Gates Work, TTL – Best Logic Yet, Ins and Outs of TTL, Flip-Flops Exposed, Memory Chips, New Cassette System Standard, Build this TVT, Using Surplus Keyboards, Morse to RTTY Converter, ASCII to Baudot via a PROM, ASCII/Baudot via PROMs, A Second Way. PLUS reprints of some of the 73 editorials on computers such as Computermainia, The Great Computer Peril, Yes, But Which Kit?, Computer Publications, Ham Computing, Postal Disaster, Programs for Sale? These are reprints from 73 gathered in one place to tie the whole works together for you. Don't miss out any longer on the fun of hobby computing and the fantastic applications of these incredible devices!

THE NEW HOBBY COMPUTERS!

This book takes it from where "Hobby Computers Are Here" leaves off, with chapters on Large Scale Integration, how to choose a microprocessor chip, an introduction to programming, low cost I/O for a computer, computer arithmetic, checking memory boards, a Baudot monitor/editor system, an audible logic probe for finding those tough problems, a ham's computer, a computer QSO machine . . . and much, much more! Everything of interest is there in one volume, ready to be enjoyed by you. Don't miss this tremendous value! Only $4.95 postpaid.

THE NEW HOBBY COMPUTERS!

Mail in coupon or phone TOLL FREE (800) 258-5473
Please send me: ____ The New Hobby Computers @ $4.95
____ Hobby Computers Are Here! @ $4.95

Total order $ _____

$____ enclosed. ☐ Cash ☐ Check ☐ Money order
Bill: ☐ Master Charge ☐ BankAmericard ☐ American Express

Card # _____ Interbank # _____
Expiration date _____ Signature _____
Name _____
Address _____
City _____ State _____ Zip _____

KILOBAUD • DEPT. HC • PETERBOROUGH NH 03458 K 4/77

Dr. Douglas Hogg
36 Calle Capistrano
Santa Barbara CA 93105

If you've been looking for some good fundamental and introductory material to the world of digital-to-analog conversion techniques then this is your lucky day! If you liked Doug's article on building a paper tape reader in last month's issue, you're going to appreciate the same practical approach he takes on this subject. — John.

Interfacing the Analog World

While we live in an analog world, our favorite toys (computers) are digital. To interface the computer to certain kinds of peripherals, we need a device which can take a digital word from the computer and convert it to an analog voltage which corresponds in some direct fashion to the value of the digital word. These devices exist and are called, appropriately enough, digital to analog converters (abbreviated DAC or D/A). The subject of this article is what these devices are, what they can do, how they do it, where to get them, and how to try them out.

What is a DAC?

A DAC assigns certain output voltages (or currents) to certain digital input values. For instance, if we have a DAC with a full scale range of 0-10 volts for an 8-bit binary

Digitial Input	Analog Output (volts)
00000000	0.000
00000001	0.039
00000010	0.078
.....	.....
.....	.....
.....	.....
01111111	4.961
10000000	5.000
10000001	5.039
.....	.....
.....	.....
.....	.....
11111110	9.922
11111111	9.961

Table 1. Voltage output corresponding to various digital input values for a DAC with an output range of 0-10 volts for an 8-bit digital input. This example shows binary coding. Other common codes are Binary Coded Decimal (BCD) and the Gray Code.

Digital Input	Analog Output (volts)
00000000	−10.00
00000001	− 9.922
.....	.....
.....	.....
.....	.....
01111111	−0.078
10000000	0.000
10000001	+ 0.078
.....	.....
.....	.....
.....	.....
11111110	+ 9.844
11111111	+ 9.922

Table 2. Voltage output versus digital input for an 8-bit DAC with a full scale output of ±10 volts.

input word, we would have the voltage output versus digital input shown in Table 1. For a digital input 0 (00000000 binary) we get 0.00 volts out. For a 1 we get .039 volts. Thus, the output voltage equals the input digital value times .039 volts.

Other DACs have outputs which might go from +10V to -10V. The common coding for these DACs is shown in Table 2. The principle is the same as in Table 1 except that 10000000 binary now gives zero volts output and each bit changes the output by .078 V. In this case, the output voltage equals the digital word minus 128, times .078 volts.

What Can DACs Be Used For?

This section suggests some of the many DAC uses, while a later section shows how to try some of these ideas out on your own computer.

The simplest application is a ramp (sawtooth) generator. If the DAC input is continuously incremented, the output will increase to the maximum and then change to zero. The faster the input is incremented (or the larger the steps), the shorter the period of the ramp will be. A fast DAC and counter can be used to generate the horizontal and vertical sweeps for a video display. Two DACs (one for the x axis and one for the y axis) are needed to position the trace on an oscilloscope in a vector graphics unit.

For applications requiring audio output, the ramp generated can be fed through an audio amplifier to a speaker. The sawtooth waveforms are suitable for alarms and such, but for longer listening, sine waves are preferable. If we store a table of values of sin(x) in memory and then sequentially output these numbers to the DAC, we can generate sine waves.

For music generation we can even add in overtones by generating a table of sin(x) + Asin(2x) + Bsin(4x), etc. In this fashion we can emulate certain instruments by varying the coefficients A, B, etc. We can also have the computer change instruments merely by changing the table of values. From this, one interesting possibility arises. Although a hobby computer is not fast enough to emulate several instruments at once, it can do a good job on one. This means that we could assemble a pretty good *orchestra* by getting several computers to *play* at the same time.

This discussion has centered on the minimum hardware way of generating music. A DAC in conjunction with an external voltage controlled oscillator (VCO) can also be used to generate tones. An additional DAC application useful here is for a digital volume control. A DAC fed into an optical isolator, whose output forms part of a voltage divider, makes a good volume control. To control the output level of a DAC connected to a computer, we can use a multiplying DAC. This type, which is the most common, gives an output proportional to the digital input, times a reference voltage. If this reference voltage comes from the output of another DAC, we can easily control the output swing of the first DAC. This combination of the two DACs can be used for voice synthesis as well as music.

Other useful applications include motor speed controls and lamp dimmers. These can be done either directly or through power amplifiers. A system of this type could make a very impressive model train controller.

The sine wave outputs are also useful for Touchtone* dialing so that your computer can call others to exchange information or complaints. This can be used for the signal source for a modem to send data on the phone lines (it will not decode the incoming information though) or to

*Trademark, Western Electric.

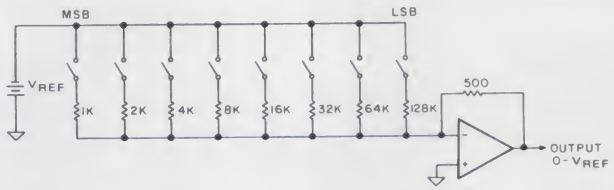


Fig. 1. A simple digital-to-analog converter circuit.

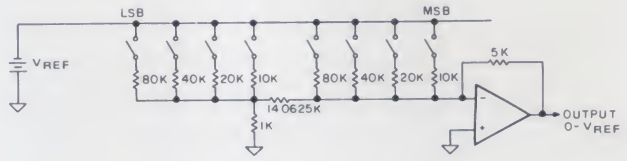


Fig. 2. An improved DAC circuit with current division between each group of four bits.

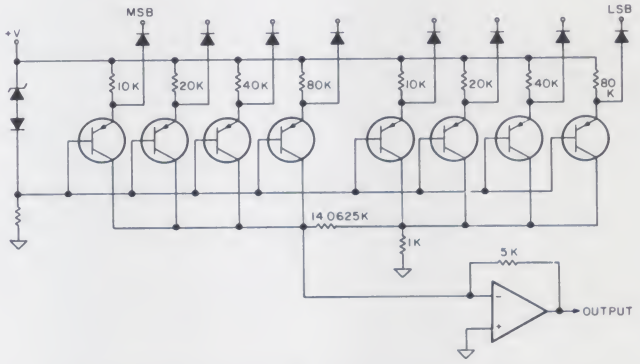


Fig. 3. A complete DAC with TTL level inputs and transistor switching.

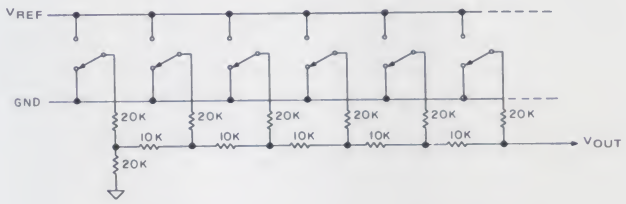


Fig. 4. A simple DAC with voltage output. This circuit, with the ground side of the switch omitted, is often used as a current source DAC.

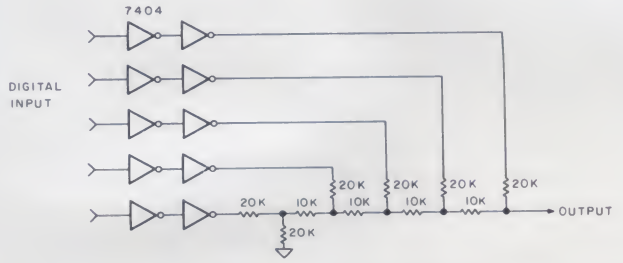


Fig. 5. A simple, practical, and quick and dirty DAC. The 10k resistors should be made from two 20k resistors in parallel. Other values of resistors may be used but they should be in the range from 4.7k to 47k and should be well matched.

Manufacturer	Model	Bits	Output	Settling Time	Power Supply (volts)	Package	Price (each)	Comments
Analog Devices	AD7520JN	10	Current	500 nsec	+5 to +15	16 pin DIP	$15.75	CMOS
	AD7521JN	12	Current	500 nsec	+5 to +15	18 pin DIP	$20.75	No internal voltage reference
Datel	DAC98BI	8	Current	500 nsec	+15	2" square	$16.95	
	DACIC8BC	8	Current	300 nsec	+5, -15	16 pin DIP	$ 8.95	No internal reference
	DACIC10BC	10	Current	250 nsec	+5, -15	16 pin DIP	$14.90	
	DACIC12B	12	Current	500 nsec	+5	18 pin DIP	NA	CMOS new product
PMI	DAC08CZ	8	Current	85 nsec	±5 to ±18	16 pin DIP	$ 9.75	IC
Motorola	1408L8	8	Current	300 nsec	NA	16 pin DIP	$ 9.00	Same as Datel DACIC8BC
Zeltex	ZD430	8	Voltage	20 msec	+5, ±15	2" square	$ 4.95*	Older design but good general purpose device

Addresses:
Analog Devices, P.O. Box 280 Norwood MA
Datel, 1020G Turnpike St., Building S., Canton MA 02021
PMI, 1500 Space Park Drive, Santa Clara CA 95050
Motorola Semiconductor, Phoenix AZ

*Available surplus from Tri-Tek, 6522 N. 43rd Ave., Phoenix AZ 85301

Table 3. Some of the available and inexpensive DACs for hobbyist applications.

cassette recorders.

These are just a few of the obvious applications. Almost any linear control function can be done with DACs.

How DACs Work

There is no one optimal DAC design. All of the designs have some trade-offs. Some are high speed, some are low power and some are low cost. The recent availability of integrated circuit DACs is bringing the ideal much closer, particularly with respect to price. Let us look at some of the methods used in the various DACs.

The simplest DAC is shown in Fig. 1. Each switch puts current into the op amp. The amount of current from each switch is determined by the binary position of that switch. The MSB (most significant bit) puts V_{ref}/1k Ohm into the op amp while the next switch puts only half that amount and so on down the line. This method is simple but for large numbers of bits the resistor for the LSB (least significant bit) gets inconveniently large. The size of this resistor not only affects the stability but also the settling time. In addition we are not going to be using switches. Rather, we will want to drive the DAC from TTL logic levels.

We can improve this method by using the circuit of Fig. 2 which attenuates the current between each group of 4 bits. This method requires only a small number of similarly valued resistors and can be extended to a large number of bits. To replace the mechanical switches we need matched transistor switches. One way of using them is shown in Fig. 3. For noncritical applications this circuit may be used as is.

The previous circuits produced current outputs which were then converted to voltages by the op amp. There is another circuit which produces a voltage output directly. This circuit is shown in Fig. 4. This R-2R ladder network can be extended to arbitrary lengths but due to resistor tolerances (.1% required by 9 bits) is not. In commercial units this type of converter also gets the same style of transistor driver shown in Fig. 3. (This circuit is most commonly used in the current mode by using exactly the drivers of Fig. 3). For home use, a quick and dirty four to six bit DAC can be made by simply connecting up the circuit shown in Fig. 5. How well this works depends on how well the logic level one and logic level zero voltages are matched between the inverters. With resistors matched to 1 or 2% (a digital ohmmeter helps, but trial and error works) a reasonable 5-bit converter can be constructed. Four bits is easy, and six bits is stretching it. Errors in matching either the resistors or inverters show up as differences in step height and are seen easiest when producing an output ramp. If the second set of inverters is replaced by CMOS inverters, only the resistors need be worried about, since the CMOS inverters are much better matched and indeed have been used in DACs with up to 9 bits with no special precautions.

There are, of course, many other ways to construct DACs but the basic ideas do not differ very much from the ones shown above.

What is Available

Table 3 is a brief listing of the most readily available inexpensive DACs. While all of the units listed are in the $20 or less range, special versions cost upwards of several hundred dollars.

In looking at the table it is good to keep in mind that not that many years ago most DACs cost hundreds, if not thousands, of dollars and it is the semiconductor industry that has brought the price of

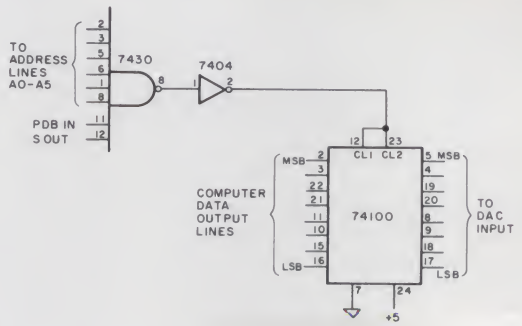


Fig. 6. A simple parallel output port for use with a DAC. This port only decodes 6 of the 8 address lines and so will respond to port numbers 63, 127, 191 and 255.

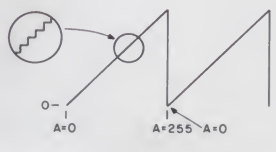


Fig. 7a. Output from a DAC with the ramp generator program discussed in the test.

Fig. 7b. Output from the DAC with ADD 32 replacing INR A.

these items down to hobbyist levels.

Exercises with a DAC

To get a feel for using the DAC, here are a few suggestions for exercises to do with one. If you have an 8-bit DAC, fine. If not, you can still follow the exercises even with the quick and dirty circuit of Fig. 5. This circuit can be simplified even more by removing the inverters and driving the resistors directly from the output port. You do, however, need a parallel output port. A port for the Altair bus is shown in Fig. 6. Something similar for the 6800 can be made but the memory mapped I/O makes it more complicated to do the address decoding.

The simplest use of the DAC is as a ramp generator. If we use the simple program below on an 8080 computer with an 8-bit DAC connected to output port 255, we will get a ramp as shown in Fig. 7a. (If you are using the quick and dirty circuit, you will not have 8 bits. Connect the five bits to the most significant bits of the output port).

Program A will:

1. Set a Counter (A) to zero.
2. Add 1 to that counter.
3. Output the number in the counter to the DAC.
4. Go back to step 2 and repeat.

Each cycle through the loop takes (with no wait states) 12.5 microseconds so the ramp cycle time is 12.5 times 256 = 3.2 μs. If we want a slower ramp we can use the modified Program B.

Program B has some additional steps between 3 and 4 which mean:

3a. Set another counter (B) to a number of my choice X.

3b. Subtract 1 from B.

3c. Is B=0? If so continue to step 4. If not got to step 3a.

The delay introduced in each cycle through the loop is X*7.5 microseconds +3.5 microseconds. For X=1 the total ramp period is increased by 2.8 μs so the total period is 6 μs. For X=2 the total period is 8.8 μs. The maximum period (X=0) is about a half a second. This can of course be extended with additional delay loops.

In some cases, transistor curve tracers for example, we would like to have either fast ramps or coarse ramps, or both. The coarse ramps are known as staircases. (The ramp we generated above is of course a staircase wave-

MEMORY LOCATION	SIN(X*3.14159/64)	(1/1.3) SIN(X*3.14/64) +(1/2.6) SIN(X*3.14/32)	(1/1.5) SIN(3.14*X/64) +(1/3) SIN(3.14*X/32) +(1/6) SIN(3.14*X/16)
000	200	200	200
004	214	223	230
010	230	246	260
014	245	267	304
020	260	310	324
024	274	327	337
030	307	344	345
034	321	356	350
040	332	366	347
044	342	374	343
050	352	377	337
054	360	377	333
060	366	375	327
064	372	371	325
070	375	363	324
074	377	353	325
100	377	342	325
104	377	330	324
110	375	315	322
114	372	302	315
120	366	270	306
124	360	255	273
130	352	244	256
134	342	233	240
140	332	224	221
144	321	216	204
150	307	211	170
154	274	205	161
160	260	202	155
164	245	201	155
170	230	200	161
174	214	200	167
200	200	200	177
204	163	177	210
210	147	177	216
214	132	176	222
220	117	175	222
224	103	172	216
230	070	166	207
234	056	161	173
240	045	153	156
244	035	144	137
250	025	133	121
254	017	122	104
260	011	107	071
264	005	075	062
270	002	062	055
274	000	047	053
300	000	035	052
304	000	024	052
310	002	014	053
314	005	006	052
320	011	002	050
324	017	000	044
330	025	000	040
334	035	003	034
340	045	011	030
344	056	021	027
350	070	033	032
354	103	050	040
360	117	067	053
364	132	110	073
370	147	131	117
374	163	154	147

Fig. 4. A simple DAC with voltage output. This circuit, with the ground side of the switch omitted, is often used as a current source DAC.

form but with 8-bit resolution it is difficult to see the individual steps). We can make a staircase generator with any number of steps we want. If we replace the INR A statement with ADD 16 the staircase will have 256/16=16 steps. ADD 32 gives 256/32=8 steps and so on (see Fig. 7b).

The output from the DAC can be fed into an audio amplifier to produce some interesting sounds, as both the period and number of steps per cycle are changes. This sort of demonstration is always useful when someone asks what the computer can do.

As usual, pleasant sounds are more soothing so we can store data shown in Table 4 in RAM to generate a sine wave, a sine wave with second harmonic or a sine wave with both second and fourth harmonic content. If the data is stored in the indicated memory locations, Program C will produce the sine waves from the DAC. The data is stored in every fourth location since the sine wave has been broken into 64 parts and we want to use the simplest program we can. If you wish, you can calculate the remaining points and modify the program to read them. This will make the maximum frequency lower and there is not much difference in sound between 64 samples and 256 samples per cycle. Do feel free to experiment.

```
        MVI A,0          076   000
LOOP    INR A            074
        OUT DAC          323   377
        JMP LOOP         303   (addr of LOOP)
```

Program A

```
        MVI A,0          076   000
LOOP    INR A            074
        OUT DAC          323   377
        MVI B,X          006   X
TIME    DCR B            005
        JNZ TIME         302   (addr of TIME)
        JMP LOOP         303   (addr of LOOP)
```

Program B

```
        LXI H, (start of data)   041   (data addr)
LOOP    MOV A,M          176
        OUT DAC          323   377
        MOV A,L          175
        ADI 4            306   004
        MOV L,A          157
        JMP LOOP         303   (addr of LOOP)
```

Program C

Again, a breakdown of what the program is doing:

1. Point to the area of memory containing the data for sine wave.
2. Move one byte of data from the memory to the accumulator.
3. Move the data from the accumulator to the DAC.
4, 5, 6. Add 4 to the lower part of the address pointer. This points to the next piece of data we want to output.
7. Go to step 2 and repeat.

To generate lower frequency notes (this generates 710Hz notes) the delay steps can be added as in the ramp generator program. To generate higher frequency notes, the ADI 4 statement can be changed to ADI 8, ADI 16, or ADI 32. This generates 1420Hz, 2840Hz or 5680Hz notes. By combining both the delay steps and the different addition constants, notes of almost any frequency can be generated. From this basic idea a fairly nice music playing program can be built up.

This short discussion of DACs does not pretend to be complete. It is an introduction and a place for you to begin experimenting. Hopefully, this will help you add a new dimension to your toy. ■

FLIP OVER OUR FLOPPY

Only $750 from Peripheral Vision.

Peripheral Vision is a brand-new company that's dedicated to selling reasonably priced peripherals for various manufacturers' CPU's.

We think you'll flip over our first product.

It's a full-size floppy disk for the Altair-Imsai plug-in compatible S-100 BUS. And it's available for as low as $750.

Here are the features:

- 1 interface card supports 4 drives
- Stores over 300,000 bytes per floppy
- Bootstrap EPROM included—no more toggling or paper tape
- Completely S-100 plug-in compatible
- Interface cabling included
- Drive is from Innovex (the originator of the floppy concept)—assembled and tested
- Interface card design is licensed from Dr. Kenneth Welles and the Digital Group
- Disk operating system with file management system included on floppy
- Cabinet and power supply optional

Prices:	Kit	Assm.
Interface card kit and assembled and tested drive	$750	$850
Power supply—+24V at 2A	45	65
Cabinet—Optima, blue	—	85

Now, a little more about our company.

Peripheral Vision may be brand-new, but we have some old-fashioned ideas about how to run our business.

We know there are serious incompatibilities among the different manufacturers' peripherals and CPU's. We want to get them together. And, we want to bring significant new products to market—products consisting of everything from adaptation instructions/kits for hardware and software to major new products.

It's a tall order, but we feel we're up to the task. Peripheral Vision has already obtained a license from The Digital Group to adapt versions of some of their products to the S-100 BUS. And we're working on getting more from other companies.

Most important to our customers, Peripheral Vision is committed to helping you get along with your computer. We'll do all we can to make it easy.

Write us now for all the information on our company, our philosophy and our exciting line of products. And be prepared to flip over all of it.

PERIPHERAL VISION

P.O. Box 6267 / Denver, Colorado 80206 / (303) 733-1678

Send me the works, and I just might flip over it!

Name_____

Address_____

City/State/Zip_____

Peter A. Stark
P.O. Box 209
Mt. Kisco NY 10549

Everything about Semiconductor Memory

... at least 4K is needed for BASIC

Pete Stark has done it again. The following contains some good introductory material to memories for the newcomer . . . and some of the neatest all-in-one-place reference material for everyone *on semiconductor memory ships. — John.*

Next to the input and output equipment, computer memories usually comprise the most expensive part of an entire computer. Until the last few years the most popular fast memory was the core memory. In the last few years, however, semiconductor advances have brought the price of integrated circuit memory to the point where IC memory is a very popular choice. Even now its price is lower than that of core memory, and with the coming of bigger and more complex ICs the price is sure to go even lower in the future.

One of the signs of the popularity of IC memory is the low price of the ICs advertised in *Kilobaud, 73 Magazine*, and elsewhere. Many different ICs are available, and sometimes it is a bit hard to figure out what all the IC numbers mean. The purpose of this article is to explain the differences between different memory ICs and present a big cross-indexing table which matches up the numbers, manufacturers, and characteristics of dozens and dozens of memory ICs.

Before continuing, remember that these memories are used to store *binary* numbers. Unlike decimal numbers which use the digits 0, 1, 2, and so on through 9, binary numbers use only 0 and 1. Each of these binary digits is called a *bit*, with every bit stored in a transistor circuit called a *cell*. More on this later.

Memory Types

Semiconductor memories come in several types depending on construction and use. These include the RAM and the ROM.

RAM is an acronym for Random-Access Memories, which are used for the temporary storage of programs and data. Numbers can be entered into the memory; this is called *writing*. Numbers can also be pulled out of memory; this is called *reading*. RAMs are designed for easy writing and reading, and are also often called read-write memories.

The word temporary should be stressed here to point out that numbers are stored in a RAM only as long as power is applied to it; in some cases we must apply not only power but must keep some timing signals coming to the RAM IC as well, to avoid loss of data. This is a weak point of semiconductor memories as compared with core memories which have the ability to retain data when the power is removed. (On the other hand, some memory ICs use so little power that it is possible to use battery power on the memory so that data is not lost even if power is turned off or fails.)

The other three types of memory are designed primarily for reading data; the letters ROM stand for Read Only Memory. ROMs are used for program instructions and data which do not change. For example, a common commercial application of microprocessor ICs is to control sophisticated cash registers, which are often called point-of-sale terminals. Such a terminal has a small computer inside which controls the operation. It would contain one or more ROMs which contain the program and fixed data, and a small RAM which would only hold changeable data such as the data pertaining to a single sale. The ROM would be written during manufacture or installation of the machine with data and instructions which would stay constant, while the data in the RAM would probably change for each sale. The use of the ROM for permanent data and programs guarantees that turning off the power or pulling the plug will not erase it.

One type of ROM is a mask-programmed ROM. This type of ROM is written into only once — during the manufacture of the IC at the semiconductor manufacturer's plant. During the manufacture, photographic negatives called masks are used to control the interconnection of the cells on the chip as well as the construction of the chip itself. If you are a big enough user, you can pay to have a special mask made just to store your specific set of numbers, so that you get a

ROM IC made especially for you. Since the mask charge is around $1000, this approach obviously is economical only if you have a need for thousands of these ICs. This puts mask-programmed ROMs out of reach of the hobbyist, but there are several of them available for specific conversion tasks, such as converting from Baudot RTTY code to ASCII. (Our memory table lists quite a number of mask-programmed ROMs just in case you run across them.)

A PROM is a ROM which can be written after manufacture. The most common PROM has a tiny nichrome wire link at each cell which acts like a fuse. By applying a high current at the right combination of pins, these fuses can be burned out one by one, with the result that some cells wind up storing a 0 and others a 1 bit. This *programming* process can only be done once so that if you make a mistake you must start fresh with a new IC. PROM (Programmed ROM) programming can be done by the manufacturer, the distributor, or by you. Programming can be done in a fairly simple jury-rigged setup, or a complicated programmer can be used which programs the ROM from data stored in a computer or on a punched paper tape or card. Commercial programmers are quite expensive, but have the advantage that they produce fewer errors and also permit easy programming of multiple chips.

There are several types of ROMs which can be erased; the Erasable Programmable ROM (EPROM) is erased by placing it under a strong ultraviolet light; the IC has a quartz glass window through which the ultraviolet can get to the chip. This type of ROM can be rewritten at any time, but this requires first erasing the entire IC and then rewriting everything. Such an EPROM is good for the initial development of a system where the program or data must be changed a number of times until the system works, but it is not suitable for frequent changes.

A fairly new type of ROM is the EAROM or RMM. The EA stands for Electrically Alterable; this is a bit like a RAM in that it can be both written and read, but the writing speed is very slow compared with reading, and has to be preceded by an erasing step which may erase an entire block of memory. Hence this type of ROM is used mostly for reading but with an occasional write; hence RMM stands for Read Mostly Memory. But the big advantage of an EAROM as opposed to a plain RAM is that the EAROM remembers data even after power is turned off; in that respect it behaves like any other ROM.

We've included a spec sheet for the WOM invented several years ago by Signetics. Unfortunately, few people managed to think of any uses for a Write-Only Memory.

Manufacturing Process

Just as any other digital IC, memories are made by a variety of manufacturing processes, each of which has different characteristics.

Bipolar ICs tend to be the fastest but consume more power per stored bit. These include the Transistor-Transistor Logic (TTL) memories which tend to have numbers beginning with 74 or 82 (Signetics has a variety of TTL ICs with 82-series numbers). ECL memories (Emitter-Coupled Logic), which are even faster, have numbers like 10142 to match other numbers in the 10,000-series ECL family. A third type of bipolar IC is the I^2L, or Integrated Injection Logic type, pioneered by Texas Instruments; only a few I^2L memory ICs are available so far.

MOSFET (Metal-Oxide Semiconductor Field Effect Transistor) ICs also come in several types — PMOS, NMOS, and CMOS. PMOS uses all P-channel FETs, while NMOS uses N-channel FETs; CMOS uses both in a Complimentary-symmetry circuit which explains the letter C in CMOS. PMOS is the older process for large complicated ICs, with NMOS becoming more popular because it is somewhat faster. PMOS and NMOS ICs often use two or even three power supplies, while CMOS uses only one; furthermore, the required voltages for PMOS and NMOS must be fairly well regulated, whereas CMOS can work on a wide variety of voltages, often ranging from about 3 up to about 15 volts.

Aside from speed (MOS ICs are slower in general than bipolar ICs), the big difference between the two is in power dissipation. MOS ICs use much less power per bit than bipolar ones; hence all the large memory ICs tend to be MOS. CMOS takes the least power — when sitting still and not being clocked or cycled, CMOS uses almost zero power, but its power dissipation goes up quickly the faster it is used. Battery operation of CMOS memories when the rest of the computer is shut down is a very practical idea.

Operating Mode

RAM memories come in basically two types — static and dynamic. The static RAM is essentially a large number of flip-flops (one per bit). Once something is written into the flip-flop it stays there as long as power is applied. This flip-flop is part of the memory cell holding that flip-flop. But each cell must also contain additional circuits to permit it to be accessed selectively, so a static memory cell can get to be quite complex. In some cases there can be five or ten transistors per cell. In a large memory this approach needs too many transistors and other components to fit into a single IC, and so static RAMs are generally limited to a thousand cells per IC or less.

To eliminate transistors, the dynamic RAM uses other components to store each bit. A common technique is to store each bit in a tiny capacitor by charging it or discharging it. Rather than build thousands of tiny capacitors into the IC, the capacitance of the FET gate lead is used. By some ingenious designing, the memory cell can be cut down to just one or two transistors, thus allowing many more bits to be stored on a single chip.

Refreshing is done by cycling through the memory IC in a certain way at a rate of about once every two milliseconds.

But since the capacitor has a tendency to discharge with time, the data has a tendency to disappear unless the capacitor is periodically recharged; this is called *refreshing*. Refreshing is done by cycling through the memory IC in a certain way at a rate of about once every two milliseconds. In a very few cases the additional circuitry to do this is on the chip itself, so that the IC looks to an outside observer as though it is static, when it is actually internally refreshing itself all the time. In most cases, however, external circuitry to refresh a dynamic RAM is needed. This complicates semiconductor memory design in many ways and creates quite a few problems. One interesting problem occurs when the computer needs to read or write into a chip just as the refreshing circuitry is busy doing its job, at which time it will just have to wait until refreshing is done.

Memory Size and Organization

The size of a memory is specified in terms of the number of bits it can store. This number of bits is usually a power of two; common memories can store 64, 256, 512, 1024, or 4096 bits. Each of these numbers is a power of two. In addition to the total number of bits that can be stored in the IC, though, we must also know just how they are stored.

We say that a memory is divided into *memory locations*, each of which can store a number (called a *word*), and each of which also has an *address*. In some cases a location only stores one bit, in which case we say that the word length is one. Other ICs may store eight bits in a location, in which case we say that the word-length is eight bits. When reading or writing is done with an IC memory, all the bits of a particular location are read or written at the *same time*. To do all of this at the same time requires that a separate IC pin be used for each bit. Thus, an IC with an eight-bit word-length would require eight pins for data to be written or read. In some cases the same eight pins may be shared for both reading and writing (this is called *multiplexing* data on the same pins), while in other cases eight pins would be used for writing and another eight for reading.

The address is used to control that location which is read or written. The more locations there are in the memory the greater is the range of possible addresses. These addresses are also binary numbers, and the greater the range of addresses the more bits are needed for the address. For example, a memory with 64 locations needs only 6 bits for an address, since 2^6 is 64. On the other hand, 4096 locations require 12-bit addresses, since 2^{12} is 4096. Usually an IC pin is required for each bit of the address, though in a few cases pins are shared so that one pin is used for two address bits.

We say that a memory is divided into memory locations, each of which can store a number (called a word), and each of which also has an address.

Assuming no pin sharing, let's look at some examples. An IC storing 256 bits can be built so that it has 256 separate locations, each holding only one bit. Whenever any reading or writing is done, only one bit is handled at a time. This IC would then need 8 bits for an address (since 2^8 is 256), one bit for writing and one bit for reading. This involves a total of ten IC pins.

On the other hand, a 256-bit IC could be built as 64 four-bit words. This requires 6 pins for an address (2^6 is 64), four pins for data input for writing, and another four pins for data output during reading. This involves a total of 14 IC pins. By sharing the same four pins for reading and writing this could be cut down to ten.

Both ICs have the same number of bits, but they are organized in different ways. The one with 256 locations would be called a 256 x 1 memory, having 256 words of 1 bit each. The one with 64 words of 4 bits each is then called 64 x 4. Note that in the 256 x 1 only one bit is written or read at a time, whereas the 64 x 4 IC reads or writes 4 bits at a time.

Now let's suppose we need a computer memory having 256 locations, each with 4 bits, for a total of 1024 bits (1024 is called a K, so this would be a memory of 1K bits). Either of the two 256-bit ICs could be used to make the memory, but in each case we would need four ICs put together to make the complete memory. But either way we must remember one thing — if the computer has a need for 4-bit numbers, it wants them one at a time. So how do we build a 1024-word memory out of four 256-word ICs?

Using the 256 x 1 ICs we would store each of the four bits in the computer's 4-bit word in a separate IC. In this way by reading or writing into all four ICs at the same time we can handle four bits at a time, one for each memory IC. A given 4-bit number is then spread out over four ICs.

Using the 64 x 4 IC we can store a 4-bit number all within one IC and access it all at once. But since it has only 64 locations we need a total of four ICs anyway. We store the first 64 memory locations in the first IC, the next 64 in the second IC, the next 64 in the third, and the last 64 in the fourth IC. For any given read or write, only one IC will be working while the other three can rest. But the one working IC will be handling four bits at a time.

In this particular case we need a way of selecting which of the four ICs is going to read or write. This is done with additional IC inputs called *chip enables* and *chip selects*. A given memory IC can have anywhere from 1 to 6 chip select pins which can be wired up in various ways to select the working IC.

The idea of memory and IC organization is a very important one to understand, so let's try still another example. Suppose we need a 256-word memory for an 8-bit word-length computer. What can we use? There are several ways we can construct this 2048-bit (2K) memory.

Eight 256 x 1 ICs will give us the total of 2048 bits. Each IC stores 256 bits which can only be accessed one at a time, so we store one bit of each of the 8-bit desired words in a different IC. All eight ICs working together (in *parallel*) can read or write 8 bits at one time.

There are many other possibilities, such as using one 256 x 8, sixteen 32 x 4 or thirty-two 64 x 1. To keep the numbers of ICs small the commercial equipment designer will use the bigger ICs, while the hobbyist may use the smaller and cheaper ICs.

For example, the designer of a very large memory, or of a memory with an odd word-length (such as, say, 9 bits or 15 bits) might use a 4096 x 1 memory since it keeps things simple and lets him tailor the word length to his needs. On the other hand, the designer of a small memory might prefer a 128 x 4 or 512 x 8, especially when some unusual number of memory locations is needed. For instance, a small microprocessor used in a cash register might need only 45 words of RAM with an 8-bit word-length. Three 16 x 8 RAMs would give him 48 words, with just a slight waste. One 64 x 8 would waste a bit more, while eight 256 x 1 RAMs would work but give him 256 words whereas he only needs 45.

Notice that he couldn't make do with two 256 x 1 RAMs even though they would give him enough bits. He needs 360 bits (45 words of 8 bits each), while two 256 x 1 RAMs would give him 512, more than he needs. But with these RAMs he could only write or read 2 bits at a time, one in each IC. To read or write a total of 8 bits he would have to repeat this four times, thus wasting a lot of time. In addition, he would need a lot of extra circuitry to convert the one address the computer gives him into four separate addresses and to also store all 8 bits temporarily while the memory ICs only handled two at a time. This is just not practical.

Inputs and Outputs

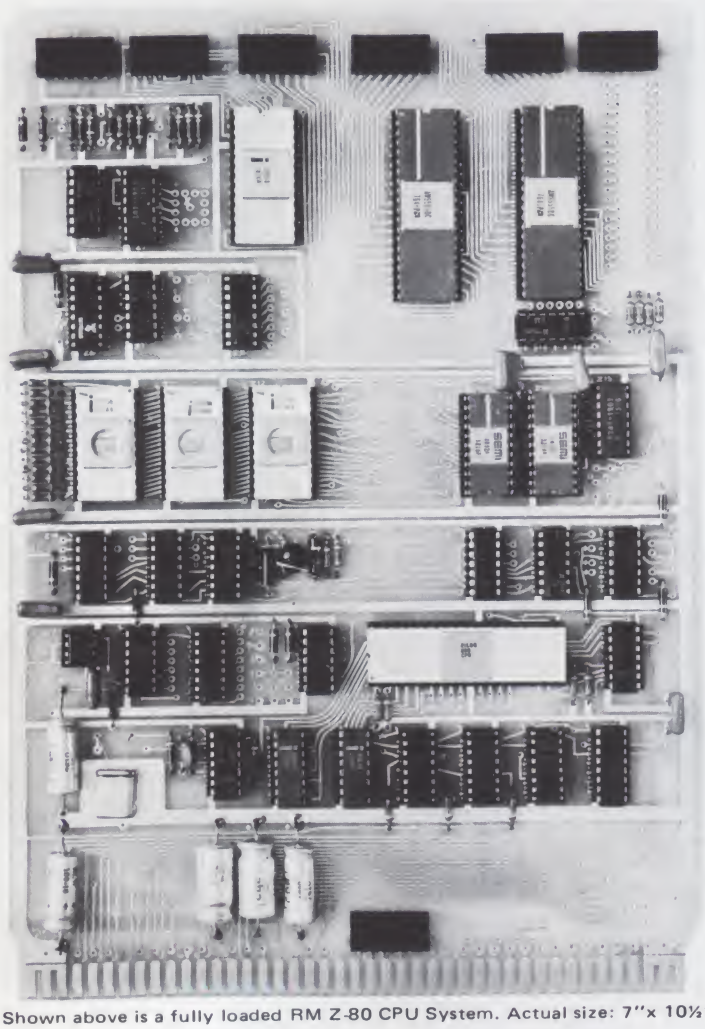
When you open up an IC and look at the chip inside,

SINGLE-BOARD or MULTI-BOARD Z-80 SYSTEM

NO SYSTEM COMPARES IN DOLLAR VALUE OR COMPACTNESS !!!

FEW CHALLENGE ITS CAPABILITY OR FLEXIBILITY

Our **RM Z-80 CPU SYSTEM** shown below, can be implemented as a stand-alone minicomputer system by merely adding a power supply and a Teletype or TV terminal as an I/O device (on-board interface included). ALONE, it stands as a fast, powerful minicomputer system. You can add memory or accessory boards at any time and when packaged in our RM Terminal case (below right), it is one of the most powerful self-contained desk-top minicomputers on the market. In addition to the CPU board, the case houses the power supply, keyboard, up to 64K of memory, a video display interface board, an optional Teletype, and special-purpose boards such as audio and digital cassette interfaces and a scientific calculator interface board. After all these, there is still room in the enclosure for your "home-brew" board built on one of several RM prototype boards available from MINI MICRO MART.



Shown above is a fully loaded RM Z-80 CPU System. Actual size: 7"x 10½".

Our **RM Z80-300 CPU SYSTEM** comes as a kit, complete with Z-80, TTL, resistors and capacitors, crystal clock, 1K of 255ns static memory and one 8255 interface IC, as well as 20-mil TTY circuitry. On-board provision exists for other parallel and serial interfaces as well as three 2708 E PROMs. All RM PC boards are epoxy glass, with plated-through holes and gold fingers. It is ..... **INTRODUCTORY PRICED AT** **$199.95**

If 1K operating monitor in a 2708 PROM is desired, order as **RM Z80-350** at **$264.95**

For all of above plus an additional RM 4K 300ns memory board, a 5-position backplane and two edge connectors, order as **RM Z80-S500** at **$419.95**

All of above and with surplus RM Terminal case shown below, which includes keyboard, power supply, fan, and card racks, order as **RM Z80-550** **$529.95**

All of above and with video display interface (no TV or monitor included), order as **RM Z80-650** **$629.95**

Power supply kit available; order as **PS-375** ..... **$ 19.95**

4K static RAM boards available from $99.95 to $139.95.

16K static memory board available; order as **RM16K-300** **$479.95**

IC socket kits included with memory boards; add additional $14.95 for IC sockets for CPU board if desired. Order as RM Z80-LP.

Any board available assembled and tested for $50 additional. Write for quotation on complete systems assembled and tested.

Enclose $1 per board for shipping, handling and insurance; $10 additional for shipping RM Terminal or Teletype printer. (Excess shipping charges are refunded.)

UNTIL APRIL 30, 1977, ALL CPU BOARD PURCHASES INCLUDE Z-80 MANUAL, LISTINGS FOR MONITOR 80, AND 5K BASIC; PRICES SUBJECT TO INCREASE AFTER APRIL 30.

INTRODUCTORY OFFER -----
Audio Cassette Interface for above system --- runs BYTE (Kansas City) standard, HITS or Tone-No-Tone format and includes cassette tape of 3K Monitor 80 operating system and 5K BASIC, including manuals and source listings. Order as **RM-ARC-375** ..... **$ 39.95**

Write for surplus catalog or S-100/Altair/IMSAI catalog. Send stamped, self-addressed envelope for specific catalog or other detailed information.

MiniMicroMart

1618 James Street, Syracuse, N.Y. 13203, Phone: (315) 422-4467

M-6

you see that the memory cells are arranged in a matrix, which is a square or rectangular area on the chip, not lined up in a neat row. Obviously there has to be some circuitry which finds the right cell or cells from the binary address you give the IC. For instance, location 56 in a memory might be in the eighth row down, seventh column from the left. The circuitry which converts from the address into the row and column number where the cell is located is called the *decoding*. Almost all ICs have the decoding circuits right on the same chip as the memory cells, somewhere off on the side of the chip; these are said to be *fully decoded* or *internally decoded*. Only a few smaller ICs are not decoded, and these are not very popular. The spec sheets call this "internal address decoding."

Another important word applied to inputs and outputs on the chip is *TTL Compatible*. Since many computers use TTL ICs for interfacing between the memory and other parts of the system, it is important to know whether the memory ICs can connect directly to the TTL logic or whether additional matching circuits are needed.

Of the bipolar ICs only TTL memories are compatible; ECL and I^2L generally require separate matching circuits. Most MOS ICs have built-in matching circuitry right on the chip so they can be called TTL compatible. But unfortunately some are more compatible than others.

The most compatible are those whose every input and *output pin* can be connected directly to TTL logic. A few ICs have all their inputs and outputs compatible except one or two — usually the chip select.

In other cases the inputs and outputs may be TTL compatible but only partly. A standard TTL IC can generally feed as many as ten *other TTL ICs*; we say that the fanout of TTL logic is ten for that reason. Although the outputs of many MOS memories can feed TTL inputs, they can often feed only one or two; hence their fanout is only one or two. In some cases they can feed only low-power TTL ICs such as 74L or 74LS.

In other cases you can feed a signal from a TTL output to the input of a MOS memory IC, but an extra resistor is needed. In any case, you have to consult the manufacturer's literature to learn about these quirks.

Another difference is in the outputs. A normal TTL output provides either a high voltage (2.4 volts and above) or a low voltage (less than .4 volts) output; these are normally called a HI and a LO. Some memory ICs have an *open-collector* output which can only provide a LO but no HI. Others have *three-state* outputs which can provide a HI, a LO, and also an open circuit condition. If the chip is not selected, its output is open-circuited and it provides no output at all. Open-collector and three-state outputs are provided to permit the connection of many ICs in parallel without their shorting each other out. As you can see from our discussion of memory size and organization, computer memory systems are very often assembled from very large numbers of ICs which share between them various pins. This would be very difficult without three-state or open-collector outputs.

Memory IC Speed

There are as many different ways of measuring the speed of memory ICs as there are manufacturers. Thus, different manufacturers may specify the speed of supposedly equivalent ICs as being different. In the attached memory IC table we give three different speeds; in each case the speed is given as the time required to accomplish some job. Thus the smaller the number the faster the IC is.

One time given is the read or write cycle time; for most memory ICs these two times are almost the same. This cycle time is the total time it takes to do a read or write, from the time you start to the time you finish. If reads or writes follow each other, then this is the time between two reads or two writes. This time can be as short as 15 nanoseconds for very fast ECL memories, or as long as 1500 or 2000 nanoseconds (1.5 or 2 microseconds) for slow MOS memories. If the read cycle time and the write cycle time are different, the write time is usually a bit longer and so the table lists the write time.

ICs in the table are listed in numerical order by the first four digits of their type number, with any letters or following numbers disregarded for ordering purposes.

Another time of interest is the read access time. Often the data being read is available before the entire read cycle is done; this is because there may be some additional tasks which have to be done in the memory before the cycle is done and the IC is ready for the next cycle. The access time is the time from the start of a read cycle to the time the output data is ready. For instance, a memory with a read cycle time of 1000 nanoseconds might have an access time of only 800 nanoseconds.

Finally, some memory ICs permit a number to be read from memory, modified, and then written back into memory at the same location, all as part of the same cycle; this is called the read-modify-write cycle, and it is usually a little longer than just a plain read or write cycle. This time is also listed in the table.

It is important to keep one thing in mind — there is a tremendous spread between ICs. If you take two ICs from the same batch, one may have an access time of 500 ns and the other 600 ns. Moreover, some bits within the same IC may be faster than others. Hence, a manufacturer's spec sheet often lists minimum, typical (or average), and maximum values for these times. Wherever possible, the memory IC table lists the maximum times given. This is for two reasons: the speed of a complete memory is really determined by the slowest IC in the bunch, and the cheap ICs available to the experimenter and hobbyist generally tend to be the slower ones. A commercial memory manufacturer may be able to design his memory system for the *typical* IC then just throw out those ICs that don't meet the typical speed; the hobbyist or experimenter can't afford that.

The IC Table

The table accompanying this article lists many different ICs from many different manufacturers. The data for the table came from various sources, and it is possible that there are errors for some ICs. Moreover, since full spec sheets were not available for all ICs, many of the IC specs are missing. And so, although I cannot guarantee that the table is correct in every respect, I have tried to make it as complete as possible by cross-checking and by examining entries to make sure they are reasonable.

ICs in the table are listed in numerical order by the first four digits of their type number, with any letters or following numbers disregarded for ordering purposes. While this may appear strange (for example, 34725

comes *before* 3515 because 3472 is smaller than 3515), it turns out to be more useful if you only know part of a memory IC number.

Letters that are part of the number have been included wherever possible except if they refer to variations in temperature range. Most ICs come in two versions — one for the so-called commercial temperature range, and one for a so-called military temperature range which includes sub-zero and near-boiling temperatures. Different manufacturers use different temperature codes; most use letters before or after the numbers. For example, Signetics puts an N in front of the number for the commercial range and an S for the military range. The table simply omits all these letters since the temperature range is usually not important to the hobbyist.

On the other hand, some manufacturers use different numbers for the different ranges. For instance the National Semiconductor DM7599 is the same as the DM8599 except for the different temperature range. With one exception, the table lists such ICs under both numbers. The exception is the 7400-series of TTL ICs which is also available with numbers starting with 54 in the military range; in this case only the 74- numbers are listed.

The table is divided into over 20 columns; the following guide will explain the meaning of each:

NUMBER is the memory IC number, ordered as explained above.

MANUF is the name of the manufacturer. In many cases several manufacturers make the same IC with slightly different numbers or different letters. These often vary in their speeds although they may otherwise be interchangeable, and so are often listed separately in the table.

TYPE gives the type of memory. RAM is random access (read-write), PROM is programmable ROM, MASK is a mask-programmable ROM, EPROM is an erasable ROM (erased by ultraviolet light), while EAROM is an electrically-alterable ROM (erased by electrical signals).

TOTAL BITS and ORG specify the total memory size in bits, as well as its organization into words and bits per word.

PROC is the manufacturing process, namely TTL, ECL, NMOS, PMOS, CMOS, or IIL. No CCD (charge-coupled devices) are included.

MODE gives the operating mode. STAT means static, while D means dynamic. A number following the D, if given, gives the refresh interval; for example, a D 2 means that the IC has to be refreshed every two milliseconds.

PINS is the number of pins.

IN/OUT tells whether data inputs and data outputs are sent over separate pins (SEP) or multiplexed on the same pins (MUX).

INPUTS columns describe the characteristics of the IC inputs. A YES under the DEC indicates the addresses are internally decoded, while a NO says they are not. A YES under TTL says that the inputs are TTL compatible. A NO indicates they are not, while SOME specifies that some inputs are and some are not.

OUTPUTS columns describe the outputs. Under TYPE is a description of how the outputting is done. The 3-ST means the output is three-state; OP C means the output is open collector (or perhaps open drain in MOS memories). TTL output compatibility is described under the TTL column.

CS applies to chip selects and gives the number of chip select inputs.

POWER SUPPLIES specify the operating voltages needed.

POWER MW gives the power in milliwatts needed in normal operation while being accessed; many ICs have a lower power mode when not being accessed.

SPEED-NANOSEC gives the operating speed; maximum times are listed wherever available. RMW means a read-modify-write cycle; RWCY is a read or write cycle; ACC is the access time. ■

signetics FULLY ENCODED, 9046 x N, RANDOM ACCESS WRITE-ONLY-MEMORY **25120**

FINAL SPECIFICATION(10)

DESCRIPTION

The Signetics 25000 Series 9046XN Random Access Write-Only-Memory employs both enhancement and depletion mode P-Channel, N-Channel, and neu(1) channel MOS devices. Although a static device, a single TTL level clock phase is required to drive the on-board multi-port clock generator. Data refresh is accomplished during CB and LH periods(11). Quadri-state outputs (when applicable) allow expansion in many directions, depending on organization.

The static memory cells are operated dynamically to yield extremely low power dissipation. All inputs and outputs are directly TTL compatible when proper interfacing circuitry is employed.

Device construction is more or less S.O.S.(2).

FEATURES

- FULLY ENCODED MULTI-PORT ADDRESSING
- WRITE CYCLE TIME 80nS (MAX. TYPICAL)
- WRITE ACCESS TIME(3)
- POWER DISSIPATION 10uW/BIT TYPICAL
- CELL REFRESH TIME 2mS (MIN. TYPICAL)
- TTL/DTL COMPATIBLE INPUTS(4)
- AVAILABLE OUTPUTS "n"
- CLOCK LINE CAPACITANCE 2pF MAX.(5)
- V_{CC} = +10V
- V_{DD} = OV ± 2%
- V_{FF} = 6.3V_{ac}(6)

APPLICATIONS

DON'T CARE BUFFER STORES
LEAST SIGNIFICANT CONTROL MEMORIES
POST MORTEM MEMORIES (WEAPON SYSTEMS)
ARTIFICIAL MEMORY SYSTEMS
NON-INTELLIGENT MICRO CONTROLLERS
FIRST-IN NEVER-OUT (FINO) ASYNCHRONOUS BUFFERS
OVERFLOW REGISTER (BIT BUCKET)

PROCESS TECHNOLOGY

The use of Signetics unique SEX(7) process yields Vth (var.) and allows the design(8) and production(9) of higher performance MOS circuits than can be obtained by competitor's techniques.

BIPOLAR COMPATIBILITY

All data and clock inputs plus applicable outputs will interface directly or nearly directly with bipolar circuits of suitable characteristics. In any event use 1 amp fuses in all power supply and data lines.

INPUT PROTECTION

All terminals are provided with slip-on latex protectors for the prevention of Voltage Destruction. (PILL packaged devices do not require protection)

SILICON PACKAGING

Low cost silicon DIP packaging is implemented and reliability is assured by the use of a non-hermetic sealing technique which prevents the entrapment of harmful ions, but which allows the free exchange of friendly ions.

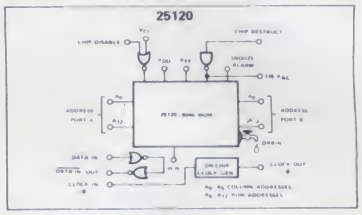
SPECIAL FEATURES

Because of the employment of the Signetics' proprietary Sanderson-Rabbet Channel the 25120 will provide 50% higher speed than you will obtain.

COOLING

The 25120 is easily cooled by employment of a six-foot fan, 1/2" from the package. If the device fails, you have exceeded the ratings. In such cases, more air is recommended.

BLOCK DIAGRAM



PART IDENTIFICATION

TYPE	"n"	TEMP. RANGE	PACKAGE
25120	0	0 to +70°C	Whatever's Right

1. "Neu" channel devices enhance or deplete regardless of gate polarity, either simultaneously or randomly. Sometimes not at all.
2. "S.O.S." copyrighted U.S. Army Commissary, 1940.
3. Not applicable.
4. You can somehow drive these inputs from TTL, the method is obvious.
5. Measure at 1MHz, 25mVac, 1.9pF in series.
6. For the filaments, what else!
7. You have a dirty mind. S.E.X. is Signetics E.Xtra Secret process. "One Shovel Full to One Shovel Full", patented by Yaguro, Kashkooli, Converse and Al. Circa 1921.
8. J. Kane calls it design (we humor him).
9. See "Modern Production Techniques" by T. Arriete (not yet written).
10. Final until we got a look at some actual parts.
11. Coffee breaks and lunch hours.
12. Due credit to EIMAC for inspiration.

NUMBER	MANUF	TYPE	TOTAL BITS	ORG WDS	BTS	PROC	MODE	PI NS	INV OUT	-INPUTS- DEC.	TTL	OUTPUTS TYPE	TTL	CS	POWER SUPPLIES	PWR MW	PWR RMW	SPEED-NANOSEC RACY	RACY ACC
SYI	SYNERTEK	RAM		1K	X 1	NMOS	STAT					3-ST			+5				150
MM4	NAT SEMI	RAM		2K	X 1	PMOS	D D					OP C			+5+8-15				420
NCM100	NITRON	MASK		512	X 4	PMOS									-12-24				800
S146	AMI	RAM		1K	X 1	PMOS	D D					OP C			19+22+5				205
NOM201C	PLESSEY	EAROM		1	X 1	NMOS									+20-20				
NOM202C	PLESSEY	EAROM		1	X 2	NMOS									+20-20				
NOM204C	PLESSEY	EAROM		1	X 4	NMOS									+20-20				
M330	SGS	RAM		1K	X 1	NMOS	STAT					3-ST			+5				
M330A	SGS	RAM		1K	X 1	NMOS	STAT					3-ST			+5				500
M330B	SGS	RAM		1K	X 1	NMOS	STAT					3-ST			+5				1000
S400	TI	RAM	4K		X	IIL	STAT	18		YES	YES		YES		+5	500			650
LS400	TI	RAM	4K		X	IIL	STAT	18		YES	YES		YES		+5	165			75
NOM401C	PLESSEY	EAROM			8 X	8	NMOS								+20				150
UPD454D	NIPPON	EAROM	2K		X		NMOS												
UPD458D	NIPPON	EAROM	8K	1K	X	8	NMOS												500
HPROM0512	HARRIS	PROM	512	64	X	8	TTL			24		YES	YES	OP C	YES 2	+5				500
SCM555D	SSS	RAM		64	X	4	CMOS	STAT						3-ST		5TO15				55
S10139	SIGNETICS	PROM		32	X	8	ECL			16		YES	NO	ECL	NO	1	-5.2			200
SN10139	TI	PROM	256	32	X	8	ECL			16		YES	NO	ECL	NO	1	-5.2			20
MMC10140	MOTOROLA	RAM		64	64	X	1	ECL	STAT	16	SEP	YES	NO	ECL	NO	1	-5.2	980		20
10140	SIGNETICS	RAM		64	64	X	1	ECL	STAT	16	SEP	YES	NO	OP C	ECL	2	-5.2			12
SN10140	TI	RAM		64	64	X	1	ECL	STAT	16	SEP	YES	NO	OP C	ECL	2	-5.2			15
SN10142	TI	RAM		64	64	X	1	ECL	STAT	16	SEP	YES	NO	OP C	ECL	2	-5.2	500		15
MCM10142	MOTOROLA	RAM		64	64	X	1	ECL	STAT	16	SEP	YES	NO	OP C	ECL	2	-5.2	500		10
MCM10143	MOTOROLA	RAM			8	X	2	ECL	STAT					OP C			-5.2			12
MMC10144	MOTOROLA	RAM			8	X	2	ECL	STAT					3-ST			-5.2			15
RC10144	RAYTHEON	RAM		256	256	X	1	ECL	STAT	16	SEP	YES	NO	OP C	ECL	3	-5.2			30
S10144	SIGNETICS	RAM		256	256	X	1	ECL	STAT	16	SEP	YES	NO	OP C	ECL	3	-5.2			35
SN10144	TI	RAM		256	256	X	1	ECL	STAT	16	SEP	YES	NO	OP C	ECL	3	-5.2	650		35
F10145C	FAIRCHILD	RAM			1K	X	1	ECL	STAT					OP C			-5.2			40
MD10145	MITACHI	RAM			16	X	4	ECL	STAT					OP C			-5.2			60
MMC10145	MOTOROLA	RAM	64	16	X	4	ECL	STAT	16		YES	NO	OP C	ECL	1	-5.2				10
S10145	SIGNETICS	RAM	64	16	X	4	ECL	STAT	16		YES	NO	OP C	ECL	1	-5.2				10
SN10145	TI	RAM	64	16	X	4	ECL	STAT	16		YES	NO	OP C	ECL	1	-5.2				15
10146	SIGNETICS	RAM		1K	X	1	ECL	STAT						OP C			-5.2	750	14	8
MMC10147	MOTOROLA	RAM	128	128	X	1	ECL	STAT	16		YES		OP C	ECL	2	-5.2				35
SN10147	TI	RAM	128	128	X	1	ECL	STAT	16		YES		OP C	ECL	2	-5.2	500			12
MMC10148	MOTOROLA	RAM	64	64	X	1	ECL	STAT	16	SEP	YES	NO	OP C	ECL	2	-5.2				15
S10148	SIGNETICS	RAM	64	64	X	1	ECL	STAT	16	SEP	YES	NO	OP C	ECL	2	-5.2				15
SN10148	TI	RAM	64	64	X	1	ECL	STAT	16	SEP	YES	NO	OP C	ECL	2	-5.2				15
MCM10149	MOTOROLA		1K	256	X	4	ECL									-5.2	500			25
10149	SIGNETICS	PROM		256	X	4	ECL							OP C		-5.2				20
10151	SIGNETICS	RAM			64	X	1	ECL	STAT					OP C		-5.2				15
HPROM1024	HARRIS	PROM	1K	256	X	4	TTL				16		YES	YES	3-ST	YES	2	+5			70
HPRM1024A	HARRIS	PROM	1K	256	X	4	TTL				16		YES	YES	OP C	YES	2	+5			70
F10405	FAIRCHILD	RAM	128	128	X	1	ECL	STAT	16		YES		OP C	ECL	2	-5.2				15
F10410	FAIRCHILD	RAM	256	256	X	1	ECL	STAT	16	SEP	YES	NO	OP C	ECL	3	-5.2				30
F10411C	FAIRCHILD	RAM			256	X	1	ECL	STAT					OP C		-4x25				35
F10413C	FAIRCHILD	RAM			1K	X	1	ECL	STAT							-4+5				30
F10415AC	FAIRCHILD	RAM			1K	X	1	ECL	STAT					OP C		-5.2				35
F10416C	FAIRCHILD	PROM			256	X	4	ECL						OP C		-5.2				50
1098-2	FAIRCHILD	RAM			4K	X	1	NMOS	D D					3-ST		+5+5+12				200
1101A	AMD	RAM			256	X	1	PMOS	STAT					3-ST		+5-9				1500
1101AM	AMD	RAM			256	X	1	PMOS	STAT					3-ST		+5-10				1500
1101A1	AMD	RAM			256	X	1	PMOS	STAT					3-ST		+5-9				1000
P1101A	INTEL	RAM	256	256	X	1	PMOS	STAT				YES		3-ST		+5-9	685	1500	1000	1500
P1101A1	INTEL	RAM	256	256	X	1	PMOS	STAT				YES		3-ST		+5-9	685	1000	1000	1000
MM1101A	NAT SEMI	RAM			256	X	1	PMOS	STAT					3-ST		+5-9				1500
MM1101A1	NAT SEMI	RAM			256	X	1	PMOS	STAT					3-ST		+5-9				1000
MM1101A2	NAT SEMI	RAM			256	X	1	PMOS	STAT					3-ST		+5-9				500
1101	SIGNETICS	RAM			256	X	1	PMOS	STAT					3-ST		+5-9				1000
S1103	AMI	RAM			1K	X	1	PMOS	D D					OP C		19+22+5				300
S1103-1	AMI	RAM			1K	X	1	PMOS	D D					OP C		19+22+5				150
S1103A	AMI	RAM			1K	X	1	PMOS	D D					OP C		16+19+5				205
S1103A-1	AMI	RAM			1K	X	1	PMOS	D D					OP C		19+22+5				145
S1103A-X	AMI	RAM			1K	X	1	PMOS	D D					OP C		19+22+5				125
S1103X	AMI	RAM			1K	X	1	PMOS	D D					OP C		19+22+5				120
1103	FAIRCHILD	RAM	1K	1K	X	1	PMOS	D	2	18	SEP	YES		OP C	NO	1	19+16	1W	580	300
1103-1	FAIRCHILD	RAM	1K	1K	X	1	PMOS	D	1	18	SEP	YES		OP C	NO	1	19+16		580	
1103F	FAIRCHILD	RAM	1K	1K	X	1	PMOS	D	2	18	SEP	YES		OP C	NO	1	19+16		270	
1103S	FAIRCHILD	RAM	1K	1K	X	1	PMOS	D	1	18	SEP	YES		OP C	NO	1	19+16		345	
P1103	INTEL	RAM	1K	1K	X	1	PMOS	D	2	18	SEP	YES		OP C	NO	1	19+16	400	580	300
1103-1	INTEL	RAM			1K	X	1	PMOS	D D					OP C			19+22+5			150
1103A	INTEL	RAM			1K	X	1	PMOS	D D					OP C			16+19+5			205
1TT1103	ITT	RAM			1K	X	1	PMOS	D D					OP C			16+19+5			300
ITT1103-1	ITT	RAM			1K	X	1	PMOS	D D					OP C			19+22+5			190
ITT1103-X	ITT	RAM			1K	X	1	PMOS	D D					OP C			19+22+5			120
MM1103	NAT SEM	RAM	1K	1K	X	1	PMOS	D	2	18	SEP	YES		OP C	NO	1	19+16			
1103	NORTEC	RAM			1K	X	1	PMOS	D D					OP C			16+19+5			300
1103-1	NORTEC	RAM			1K	X	1	PMOS	D D					OP C			19+22+5			150
1103-146	NORTEC	RAM			1K	X	1	PMOS	D D					OP C			19+22+5			205
1103A-	ROCKWELL	RAM			1K	X	1	PMOS	D D					OP C			19+22+5			110
1103A	ROCKWELL	RAM			1K	X	1	PMOS	D D					OP C			16+19+5			205
1103	SIGNETICS	RAM	1K	1K	X	1	PMOS	D	2	18	SEP	YES		OP C	NO	1	19+16	400	580	300
1103-1	SIGNETICS	RAM	1K	1K	X	1	PMOS	D	1	18	SEP	YES		OP C	NO	1	19+16	400	340	150

NUMBER	MANUF	TYPE	TOTAL BITS	ORG WDS	BTS	PROC	MODE	PI NS	INV OUT	-INPUTS- DEC.	TTL	OUTPUTS TYPE	TTL	CS	POWER SUPPLIES	PWR MW	PWR RMW	SPEED-NANOSEC RACY	RACY ACC
2102F	FAIRCHILD	RAM		1K	X 1	NMOS	STAT					3-ST	YES	1	+5		350	350	350
2102F2	FAIRCHILD	RAM		1K	X 1	NMOS	STAT					3-ST			+5				250
21L02	FAIRCHILD	RAM	1K	1K	X 1	NMOS	STAT	16	SEP	YES	YES	3-ST	YES	1	+5	158		650	650
21L02A	FAIRCHILD	RAM	1K	1K	X 1	NMOS	STAT	16	SEP	YES	YES	3-ST	YES	1	+5	158		500	500
21L02B	FAIRCHILD	RAM	1K	1K	X 1	NMOS	STAT	16	SEP	YES	YES	3-ST	YES	1	+5	158		400	400
P2102A	INTEL	RAM	1K	1K	X 1	NMOS	STAT	16	SEP	YES	YES	3-ST	YES	1	+5	350		1000	1000
2102-1	INTEL	RAM		1K	X 1	NMOS	STAT					3-ST			+5				500
2102-2	INTEL	RAM		1K	X 1	NMOS	STAT					3-ST			+5				650
2102-8	INTEL	RAM		1K	X 1	NMOS	STAT					3-ST			+5				1500
2102A	INTEL	RAM		1K	X 1	NMOS	STAT					3-ST			+5				350
P2102A-2	INTEL	RAM	1K	1K	X 1	NMOS	STAT	16	SEP	YES	YES	3-ST	YES	1	+5		250		250
P2102A-4	INTEL	RAM	1K	1K	X 1	NMOS	STAT	16	SEP	YES	YES	3-ST	YES	1	+5	350		450	450
MM2102	NAT SEMI	RAM	1K	1K	X 1	NMOS	STAT	16	SEP	YES	YES	3-ST	YES	1	+5	350		1000	1000
MM2102-1	NAT SEMI	RAM		1K	X 1	NMOS	STAT					3-ST			+5				500
MM2102-2	NAT SEMI	RAM		1K	X 1	NMOS	STAT					3-ST			+5				650
2102	SIGNETICS	RAM	1K	1K	X 1	NMOS	STAT	16	SEP	YES	YES	3-ST	YES	1	+5			1000	1000
2102-1	SIGNETICS	RAM		1K	X 1	NMOS	STAT					3-ST			+5				500
2102-2	SIGNETICS	RAM		1K	X 1	NMOS	STAT					3-ST			+5				650
21F02	SIGNETICS	RAM	1K	1K	X 1	NMOS	STAT	16	SEP	YES	YES	3-ST	YES	1	+5			350	350
21F02-2	SIGNETICS	RAM		1K	X 1	NMOS	STAT					3-ST			+5				250
21F02-4	SIGNETICS	RAM		1K	X 1	NMOS	STAT					3-ST			+5				450
21L02	SIGNETICS	RAM		1K	X 1	NMOS	STAT					3-ST			+5				1000
21L02-1	SIGNETICS	RAM	1K	1K	X 1	NMOS	STAT	16	SEP	YES	YES	3-ST	YES	1	+5			500	500
21L02-2	SIGNETICS	RAM		1K	X 1	NMOS	STAT					3-ST			+5				650
21L02-3	SIGNETICS	RAM	1K	1K	X 1	NMOS	STAT	16	SEP	YES	YES	3-ST	YES	1	+5			400	400
SY2102	SYNERTEK	RAM		1K	X 1	NMOS	STAT					3-ST			+5				300
SY2102-6	SYNERTEK	RAM		1K	X 1	NMOS	STAT					3-ST			+5				1000
SY2102A	SYNERTEK	RAM		1K	X 1	NMOS	STAT					3-ST			+5				275
SY21M02	SYNERTEK	RAM		1K	X 1	NMOS	STAT					3-ST			+5				350
SY21M02-1	SYNERTEK	RAM		1K	X 1	NMOS	STAT					3-ST			+5				175
SY21M02-2	SYNERTEK	RAM		1K	X 1	NMOS	STAT					3-ST			+5				150
SY21L02	SYNERTEK	RAM		1K	X 1	NMOS	STAT					3-ST			+5				200
SY21L02-1	SYNERTEK	RAM		1K	X 1	NMOS	STAT					3-ST			+5				1000
SY21L02B	SYNERTEKL	RAM		1K	X 1	NMOS	STAT					3-ST			+5				500
S2103	AMI	RAM	1K	1K	X 1		D								+5				400
MM2104	MITACHI	RAM		256	X 1	ECL	STAT					OP C			-5.2				40
2104	INTEL	RAM		4K	X 1	NMOS	D D					3-ST			+5+12-5				350
2104-2	INTEL	RAM		4K	X 1	NMOS	D D					3-ST			+5+12-5				250
2104-4	INTEL	RAM		4K	X 1	NMOS	D D					3-ST			+12+5-5				300
MM2105	MITACHI	RAM		256	X 1	ECL	STAT					OP C			-5+2				35
2105	INTEL	RAM		1K	X 1	NMOS	D D					OP C			-5+12				95
2105-1	INTEL	RAM		1K	X 1	NMOS	D D					OP C			-5+12				80
2105-2	INTEL	RAM		1K	X 1	NMOS	D D					OP C			-5+12				85
S2107	INTEL	RAM	4K	4K	X 1		D	22		YES					+12+5-5	460		700	300
2107A	INTEL	RAM		4K	X 1	NMOS	D D					3-ST			+12+5-5				300
C2107A-1	INTEL	RAM	4K	4K	X 1	NMOS	D		22	SEP	YES	SOME	3-ST	YES	1	+12+5-5	516	550	280
C2107A-4	INTEL	RAM	4K	4K	X 1	NMOS	D		2	SEP	YES	SOME	3-ST	YES	1	+12+5-5	405	840	350
C2107A-5	INTEL	RAM	4K	4K	X 1	NMOS	D		22	SEP	YES	SOME	3-ST	YES	1	+12+5-5	376	970	420
C2107A-8	INTEL	RAM	4K	4K	X 1	NMOS	D		22	SEP	YES	SOME	3-ST	YES	1	+12+5-5	376	970	420
C2107B	INTEL	RAM	4K	4K	X 1	NMOS	D		22	SEP	YES	SOME	3-ST	YES	1	+12+5-5	648	400	200
2107B-2	INTEL	RAM		4K	X 1	NMOS	D D					3-ST			+5+12-5				220
C2107B-4	INTEL	RAM	4K	4K	X 1	NMOS	D		22	SEP	YES	SOME	3-ST	YES	1	+12+5-5	648	470	270
2107B-6	INTEL	RAM		4K	X 1	NMOS	D D					3-ST			+5+12-5				350
2108	SIGNETICS	RAM	1K	1K	X 1		STAT												
2111	AMD	RAM			256	X 4	NMOS	STAT							3-ST				1000
2111-2	AMD	RAM			256	X 4	NMOS	STAT							3-ST				650
2111-1	AMD	RAM			256	X 4	NMOS	STAT							3-ST				500
EA2111M	ELEC ARR	RAM			256	X 4	NMOS	STAT							3-ST				500
EA2111	ELEC ARR	RAM			256	X 4	NMOS	STAT							3-ST				350
2111	INTEL	RAM	1K	256	X 4	NMOS	STAT	18	MUX	YES	YES	3-ST	YES	2	+5	350		1000	1000
2111-1	INTEL	RAM	1K	256	X 4	NMOS	STAT	18	MUX	YES	YES	3-ST	YES	2	+5	330		650	650
2111-2	INTEL	RAM	1K	256	X 4	NMOS	STAT	18	MUX	YES	YES	3-ST	YES	2	+5	350		500	500
MM2111	NAT SEMI	RAM			256	X 4	NMOS	STAT							3-ST				1000
MM2111-1	NAT SEMI	RAM			256	X 4	NMOS	STAT							3-ST				500
MM2111-2	NAT SEMI	RAM			256	X 4	NMOS	STAT							3-ST				650
2111	SIGNETICS	RAM	1K	256	X 4	NMOS	STAT	18	MUX	YES	YES	3-ST	YES	2	+5			1000	1000
2111-1	SIGNETICS	RAM	1K	256	X 4	NMOS	STAT	18	MUX	YES	YES	3-ST	YES	2	+5			500	500
2111-2	SIGNETICS	RAM	1K	256	X 4	NMOS	STAT	18	MUX	YES	YES	3-ST	YES	2	+5			650	650
SY2111	SYNERTEK	RAM			256	X 4	NMOS	STAT							3-ST				1000
SY2111-1	SYNERTEK	RAM			256	X 4	NMOS	STAT							3-ST				500
SY2111-2	SYNERTEK	RAM			256	X 4	NMOS	STAT							3-ST				650
SY2111A	SYNERTEK	RAM			256	X 4	NMOS	STAT							3-ST				350
SY2111A-2	SYNERTEK	RAM			256	X 4	NMOS	STAT							3-ST				250
SY2111A-4	SYNERTEK	RAM			256	X 4	NMOS	STAT							3-ST				450
SY21M12	SYNERTEK	RAM			256	X 4	NMOS	STAT							3-ST				175
SY21M12-1	SYNERTEK	RAM			256	X 4	NMOS	STAT							3-ST				150
SY21M12-2	SYNERTEK	RAM			256	X 4	NMOS	STAT							3-ST				200
SY2113	SYNERTEK	RAM			256	X 4	NMOS	STAT							3-ST				500
2115	INTEL	RAM			1K	X 1	NMOS	STAT							OP C				90
2115-2	INTEL	RAM			1K	X 1	NMOS	STAT							OP C				70
2115-4	INTEL	RAM			1K	X 1	NMOS	STAT							OP C				120
2115L	INTEL	RAM			1K	X 1	NMOS	STAT							OP C				95
2116	INTEL	RAM	16K	16K	X 1	NMOS	D	2	16			YES							
EA2121M	ELEC ARR	RAM			256	X 4	NMOS	STAT							3-ST		YES	0	+5+5+12
2125	INTEL	RAM			1K	X 1	NMOS	STAT							3-ST				400
2125-2	INTEL	RAM			1K	X 1	NMOS	STAT							3-ST				90
2125-4	INTEL	RAM			1K	X 1	NMOS	STAT							3-ST				70
2125L	INTEL	RAM			1K	X 1	NMOS	STAT							3-ST				120
S2146	AMI	RAM	1K	1K	X 1		D												95
2170	MMI	RAM			4K	X 1	NMOS	D D							3-ST				
2180	MMI	RAM			4K	X 1	NMOS	D D							3-ST				350
S2222	AMI	RAM			512	X 1	CMOS	STAT											350
S2222A	AMI	RAM			512	X 1	CMOS	STAT											200
2222	NORTEC	RAM			512	X 1	CMOS	STAT											400
2222A	NORTEC	RAM			512	X 1	CMOS	STAT											200
MK2300	MOSTEK	MASK			320	X 8	PMOS												400
TMS2300	TI	MASK			256	X 10	PMOS												1000
EA2308A	ELEC ARR	MASK		1K	X 8	NMOS													550
P2308	INTEL	MASK	8K	1K	X 8	NMOS													200
P2316A	INTEL	MASK	16K	2K	X 8	NMOS													500
SY2316A	SYNERTEK	MASK	16K		X							YES		YES					850
SY2316B	SYNERTEK	MASK	16K		X							YES		YES					850
MK2400	MOSTEK	MASK			256	X 10	PMOS												450
2400	N C R	EAROM			1K	X 1	NMOS												600
ER2401	GEN INST	EAROM	4K		1K	X 4	NMOS												
MK2500	MOSTEK	MASK			512	X 8	PMOS												2000
MK2500	MOSTEK	MASK			1K	X 8	PMOS												700
25L01	SIGNETICS	RAM	256	256	X 1	PMOS	STAT	16	SEP	YES									700
MM2502	MITACHI	RAM			16	X 4	TTL	STAT							OP C				1000
MM2505	MITACHI	RAM			256	X 1	TTL	STAT							3-ST				60
MM2510	MITACHI	RAM			1K	X 1	TTL	STAT							OP C				65
2530	SIGNETICS	MASK	4K	512	X 8	PMOS		24				YES			3-ST	YES	2	+5+12	70
UA2556	SOLITRON	RAM			256	X 1	PMOS	STAT											700
2580	SIGNETICS	MASK	8K	2K	X 4	NMOS						YES			3-ST	YES	4	+5+5	1000
MK2600	MOSTEK	MASK		512	X 8	PMOS									3-ST			+5+12	950
MK2600	MOSTEK	MASK			1K	X 8	PMOS								3-ST			+5+12	700
2601	SIGNETICS	RAM	1K	256	X 4	NMOS	STAT	22	SEP	YES	YES	3-ST	YES	2	+5			+5+12	700
2601-1	SIGNETICS	RAM		256	X 4	NMOS	STAT					3-ST			+5			400	400
2602	SIGNETICS	RAM	1K	1K	X 1	NMOS	STAT	16	SEP	YES	YES	3-ST	YES	1	+5			+5	500
2602-1	SIGNETICS	RAM	1K	1K	X 1	NMOS	STAT	16	SEP	YES	YES	3-ST	YES	1	+5	350		1000	1000
2602-2	SIGNETICS	RAM		1K	X 1	NMOS	STAT					3-ST			+5	350		450	450
2604	SIGNETICS	RAM		4K	X 1	NMOS	D D					3-ST			+5				650
2606	SIGNETICS	RAM	1K	256	X 4	NMOS	STAT		MUX		YES	3-ST	YES	1	+12+5-5				300
2606-1	SIGNETICS	RAM	1K	256	X 4	NMOS	STAT		MUX		YES			YES	+5			750	750
2608	SIGNETICS	MASK	8K	1K	X 8	NMOS		24		YES	YES	3-ST	YES	1	+5			900	500
2608-1	SIGNETICS	MASK	8K	1K	X 8	NMOS		24		YES	YES	3-ST	YES	4	+5				550
2611	SIGNETICS	RAM	1K	256	X 4	NMOS	STAT	18	MUX	YES	YES	3-ST	YES	2	+5				
2670-1	SIGNETICS	RAM		4K	X 1	NMOS	D D					3-ST			+12-5			400	400
2670-2	SIGNETICS	RAM		4K	X 1	NMOS	D D					3-ST			+12-5				250
2670-3	SIGNETICS	RAM		4K	X 1	NMOS	D D					3-ST			+12-5				300
2680	SIGNETICS	RAM	4K	4K	X 1	NMOS	D	2	22	SEP									
2680-1	SIGNETICS	RAM	4K	4K	X 1	NMOS	D	2	22	SEP						1	+5+12-5	520	400
2680-2	SIGNETICS	RAM	4K	4K	X 1	NMOS	D	2	22	SEP						1	+5+12-5	990	470
2700	AMD	RAM	256	256	X 1	TTL	STAT	16	SEP	YES	YES	3-ST	YES	3	+5	1	+5+12-5	960	800
27L500AC	AMD	RAM			256	X 1	TTL	STAT							3-ST				
27L500C	AMD	RAM			256	X 1	TTL	STAT							3-ST				36
AM27LS00M	AMD	RAM			256	X 1	TTL	STAT							3-ST				45
AM2701	AMD	RAM	256	256	X 1	TTL	STAT	16	SEP	YES	YES	OP C	YES	3	+5				55
27L501C	AMD	RAM			256	X 1	TTL	STAT							3-ST				
AM27LS01M	AMD	RAM			256	X 1	TTL	STAT							OP C				45
AM2502	AMD	RAM	64	16	X 4	TTL	STAT	16	SEP	YES	YES	OP C	YES	1	+5				55
AM2503	AMD	RAM	64	16	X 4	TTL	STAT	16	SEP	YES	YES	3-ST	YES	1	+5				35
C2704	INTEL	EAROM	4K	512	X 8	NMOS									3-ST				35
C2704-5	INTEL	EAROM	4K	512	X 8	NMOS									3-ST				500
AM2708	AMD	EPROM	8K	1K	X 8										+12+5-5	750			1000
AM27LS08C	AMD	PROM			32	X 8	TTL												450
AM27LS08M	AMD	PROM			32	X 8	TTL								OP C				90
27S08	AMD	PROM	256	32	X 8	TTL									OP C				75
2708	INTEL	EPROM	8K	1K	X 8	NMOS									+5				
2708-5	INTEL	EPROM			1K	X 8	NMOS								3-ST				450
															+5+5+12				1000

NUMBER	MANUF	TYPE	TOTAL BITS	ORG WDS	BTS	PROC	MODE	PI NS	INV OUT	-INPUTS- DEC.	TTL	OUTPUTS TYPE	TTL	CS	POWER SUPPLIES	PWR MW	PWR RMW	SPEED-NANOSEC RACY	RACY ACC
MM2112-2	NAT SEMI	RAM		256	X 4	NMOS	STAT					3-ST			+5				650

NUMBER	MANUF	TYPE	TOTAL BITS	ORG WDSXBTS	PROC	MODE	PI NS	I/N/ OUT	-INPUTS- DEC.	TTL	OUTPUTS TYPE	TTL	CS	POWER SUPPLIES	PWR MW	SPEED=NANOSEC RMW	RECV	ACC
AM27LS09C	AMD	PROM	256	32 X 8	TTL		16		YES	YES	3=ST	YES	1	+5				55
AM27LS09M	AMD	PROM		32 X 8	TTL						3=ST			+5				75
AM27LS10	AMD	PROM		256 X 4	TTL						OP C			+5				60
27S10	AM	PROM	1K	256 X 4	TTL		16		YES	YES	OP C	YES	2	+5				
AM27LS11	AMD	PROM		256 X 4	TTL						3=ST			+5				60
27S11	AM	PROM	1K	256 X 4	TTL		16		YES	YES	3=ST	YES	2	+5				
AM27S40	AMD	MASK		512 X 8	TTL						OP C			+5				150
AM27S40*	AMD	MASK		512 X 8	TTL						OP C			+5				180
AM27S41	AMD	MASK		512 X 8	TTL						3=ST			+5				150
AM27S41*	AMD	MASK		512 X 8	TTL						3=ST			+5				180
AM27S80	AMD	MASK		1K X 8	TTL						OP C			+5				150
AM27S81	AMD	MASK		1K X 8	TTL						3=ST			+5				150
MK28000	MOSTEK	MASK		2K X 8	PMOS									+5=12				600
MK28000	MOSTEK	MASK		4K X 4	PMOS									+5=12				800
ER2800	GEN INST	EAROM	8K	X													20MS	2000
MK29000	MOSTEK	MASK		2K X 8	NMOS						OP C			+12				250
MK29000	MOSTEK	MASK		4K X 4	NMOS						OP C			+12				250
CRC3001	COLLINS	MASK		128 X 8	PMOS									-12-27				4000
CRC3002	COLLINS	MASK		256 X 8	PMOS									+12=12				1500
CRC3002	COLLINS	MASK		512 X 4	PMOS									+12-12				1500
MK31000	MOSTEK	MASK		2K X 8	NMOS						3=ST			+5				600
3101	AMD	RAM		16 X 4	TTL	STAT					OP C			+5				60
31L01	AMD	RAM		16 X 4	TTL	STAT					OP C			+5				110
31013	AMD	RAM		16 X 4	TTL	STAT					OP C			+5				75
P3110	INTEL	RAM	64	16 X 4	TTL	STAT	16	SEP	YES	YES	OP C	YES	1	+5	525		60	60
P3101A	INTEL	RAM	64	16 X 4	TTL	STAT	16	SEP	YES	YES	OP C	YES	1	+5	525		35	35
M3101A	INTEL	RAM		16 X 4	TTL	STAT					OP C			+5				45
M3101	INTEL	RAM		16 X 4	TTL	STAT					OP C			+5				75
3101A	SIGNETICS	RAM	64	16 X 4	TTL	STAT	16		YES	YES	OP C	YES	1	+5	400			50
5310IA	SIGNETICS	RAM		16 X 4	TTL	STAT					OP C			+5				90
53102	AMI	RAM	1K	1K X 1	NMOS	STAT	16	SEP	YES	YES	3=ST	YES	1	+5	350		1000	1000
53102A	AMI	RAM	1K	1K X 1	NMOS	STAT	16	SEP	YES	YES	3=ST	YES	1	+5	350		450	450
53102B	AMI	RAM	1K	1K X 1	NMOS	STAT	16	SEP	YES	YES	3=ST	YES	1	+5	350		650	650
53103	AMI	RAM	1K	1K X 1		D												
3104	INTEL	RAM		4 X 4	TTL	STAT					OP C			+5				30
P3106	INTEL	RAM	256	256 X 1	TTL	STAT	16	SEP	YES	YES	3=ST	YES	3	+5	650		80	80
P3106A	INTEL	RAM	256	256 X 1	TTL	STAT	16	SEP	YES	YES	3=ST	YES	3	+5	650		70	60
P3106-8	INTEL	RAM	256	256 X 1	TTL	STAT	16	SEP	YES	YES	3=ST	YES	3	+5	650		80	80
P3107	INTEL	RAM	256	256 X 1	TTL	STAT	16	SEP	YES	YES	OP C	YES	3	+5	650		80	80
P3107A	INTEL	RAM	256	256 X 1	TTL	STAT	16	SEP	YES	YES	OP C	YES	3	+5	650		70	60
P3107-8	INTEL	RAM	256	256 X 1	TTL	STAT	16	SEP	YES	YES	OP C	YES	3	+5	650		70	60
3110	INTEL	RAM	1K	1K X 1	TTL	STAT	16	SEP	YES	YES	OP C	YES	1	+5				
R03=16384	GEN INST	MASK		4K X 4	NMOS						3=ST			+5				1000
R03=16384	GEN INST	MASK		4K X 4	NMOS						3=ST			+5				1000
RA3=1801	GEN INST	RAM		1K X 1	NMOS	STAT					3=ST			+15=3.5				90
RA3=1802	GEN INST	RAM		1K X 1	NMOS	STAT								+13=3.5				90
T3181A	TOSHIBA	EPROM		512 X 4	PMOS						3=ST			+5=5=9				1000
T3181B	TOSHIBA	EPROM		512 X 4	PMOS						3=ST			+3=5=9				1500
R03=2560	EMM	MASK		512 X 5	NMOS						3=ST			+5				490
R03=2560	GEN INST	RAM		512 X 5	NMOS	STAT					3=ST			+5				490
P3301A	INTEL	MASK	1K	256 X 4	TTL			16		YES	OP C	YES	2	+5	657			45
M3301A	INTEL	MASK		256 X 4	TTL									+5				60
M3301A	INTEL	MASK		256 X 4	TTL						OP C			+5				90
D3302	INTEL	MASK	2K	512 X 4	TTL						OP C			+5	735			70
D3302=4	INTEL	MASK	2K	512 X 4	TTL						OP C			+5	735			90
D3302L=8	INTEL	MASK	2K	512 X 4	TTL						OP C			+5	980			90
P3304A	INTEL	MASK	4K	512 X 8	TTL						OP C			+5	998			70
P3304A=4	INTEL	MASK	4K	512 X 8	TTL						OP C			+5	998			90
3304AL=8	INTEL	MASK		512 X 8	TTL						OP C			+5				120
P3322	INTEL	MASK	2K	512 X 4	TTL						3=ST			+5	735			70
P3322=4	INTEL	MASK	2K	512 X 4	TTL						3=ST			+5	735			90
P3322L=8	INTEL	MASK	2K	512 X 4	TTL						3=ST			+5	980			90
P3224A	INTEL	MASK	4K	512 X 8	TTL						3=ST			+5	998			70
P3224A=4	INTEL	MASK	4K	512 X 8	TTL						3=ST			+5	998			90
ER3400	GEN INST	EAROM	4K	X														650
MK34000	MOSTEK	MASK	16K	2K X 8					YES	YES	3=ST	YES	3	+5	330			350
R03=4096	EMM	MASK		512 X 8	NMOS						3=ST			+5				500
R03=4096	GEN INST	MASK		512 X 8	NMOS						3=ST			+5				500
RA3=4256	EMM	RAM		256 X 4	NMOS	STAT					3=ST			+5				1000
RA3=4256A	EMM	RAM		256 X 4	NMOS	STAT					3=ST			+5				650
RA3=4256B	EMM	RAM		256 X 4	NMOS	STAT					3=ST			+5				650
RA3=4256	GEN INST	RAM		256 X 4	NMOS	STAT					3=ST			+5				650
RA3=4402	GEN INST	RAM		4K X 1	NMOS	STAT					3=ST			-				200
34720	FAIRCHILD	RAM	256	256 X 1	CMOS	STAT	16		YES	NO	3=ST	NO	1	+5 TO +15	1			250
34725	FAIRCHILD	RAM	64	16 X 4	CMOS	STAT	16		YES		3=ST		1	+5 TO +15	0.2			200
CRC3901	COLLINS	MASK		256 X10	PMOS									+5=12				800
CRC3901	COLLINS	MASK		512 X 5	PMOS									+5=12				800
MM3501	NAT SEMI	MASK		128 X 8	PMOS									-13=27				4000
NM3503=1	HITACHI	RAM		1K X 1	PMOS	D O					OP C			19=22.5				150
HM3503=2	HITACHI	RAM		1K X 1	PMOS	O O					OP C			-				220
R03=5120	EMM	MASK		512 X10	NMOS						3=ST			+5				500
R03=5120	GEN INST	MASK		512 X10	NMOS						3=ST			+5				500
53514	AMI	MASK		512 X 8	PMOS									+5=12				1000
1T73514=1	ITT	MASK		512 X 8	PMOS									+5=12				600
1T73514=2	ITT	MASK		512 X 8	PMOS									+5=12				1000
SY3514	SYNERTEK	MASK		512 X 8	NMOS									+5				500

NUMBER	MANUF	TYPE	TOTAL BITS	ORG WDSXBTS	PROC	MODE	PI NS	I/N/ OUT	-INPUTS- DEC.	TTL	OUTPUTS TYPE	TTL	CS	POWER SUPPLIES	PWR MW	SPEED=NANOSEC RMW	RECV	ACC
TMS4043-2	TI	RAM	1K	256 X 4	NMOS	STAT	16	MUX	YES	YES	3=ST	YES	1	+5		175		450	450
MCM4046L	MOTOROLA	RAM		16 X 4	TTL	STAT					OP C			+5					60
MM4050V2	RCA	RAM		4K X 1	NMOS	D D					3=ST			+12=5					200
TMS4050	TI	RAM	4K	4K X 1	NMOS	D 2	18	MUX		SOME	OP C	YES	1	12=5	420	730	470	300
TMS4050=1	TI	RAM	4K	4K X 1	NMOS	D 2	18	MUX		SOME	OP C	YES	1	12=5	420	600	430	250
TMS4050=2	TI	RAM	4K	4K X 1	NMOS	D 2	18	MUX		SOME	OP C	YES	1	12=5	420	600	400	200
TMS4051	TI	RAM	4K	4K X 1	NMOS	D 2	18	MUX	YES	YES	OP C	YES	1	12=5	460	730	470	300
TMS4051=1	TI	RAM	4K	4K X 1	NMOS	D 2	18	MUX	YES	YES	OP C	YES	1	12=5	460	860	430	250
EA4060	ELEC ARR	RAM		4K X 1	NMOS	D O					3=ST			+5=5=12					200
RA4060=1	ELEC ARR	RAM		4K X 1	NMOS	D D					3=ST			+5=12=5					250
EA4060=2	ELEC ARR	RAM		4K X 1	NMOS	D D					3=ST			+12=5=5					300
MM4060	RCA	RAM		4K X 1	NMOS	D D					3=ST			+12=5=5					300
MM4060V1	RCA	RAM		4K X 1	NMOS	D D					3=ST			+12=5=5					270
MM4060V2	RCA	RAM		4K X 1	NMOS	D D					3=ST			+12=5=5					200
TMS4060	TI	RAM	4K	4K X 1	NMOS	D	22		YES	SOME				1		400	710	470	300
TMS4060=1	TI	RAM	4K	4K X 1	NMOS	D	22		YES	SOME				1		400	640	430	250
TMS4060-2	TI	RAM	4K	4K X 1	NMOS	D	22		YES	SOME				1		400	580	400	200
TMS4062	TI	RAM	1K	1K X 1	PMOS	D 2	22		YES	NO			NO	1 +7=20+22		120		200	130
TMS4063	TI	RAM	1K	1K X 1	PMOS	D 2	18		YES	NO			NO	1 +7=20+22		120		200	130
TMS4070	TI	RAM	4K			D	16												
S4096	AMI	RAM		4K X 1	NMOS	D D					3=ST			+12=5=5					510
S4096=1	AMI	RAM		4K X 1	NMOS	D D					3=ST			+5=12=5					350
S4096=2	AMI	RAM		4K X 1	NMOS	D D					3=ST			+12=5=5					300
S4096=3	AMI	RAM		4K X 1	NMOS	D D					3=ST			+5=12=5					250
4096	FAIRCHILD	RAM	4K	4K X 1	NMOS	D	16		SOM	YES	3=ST	YES	1	+12=5=5					
4096-2	FAIRCHILD	RAM	4K	4K X 1	NMOS	D	16		SOM	YES	3=ST	YES	1	+12=5=5	431		300	200
4096=3	FAIRCHILD	RAM	4K	4K X 1	NMOS	D	16		SOM	YES	3=ST	YES	1	+12=5=5	378		360	250
4096=4	FAIRCHILD	RAM	4K	4K X 1	NMOS	D	16		SOM	YES	3=ST	YES	1	+12=5=5	341		420	300
4096-5	FAIRCHILD	RAM	4K	4K X 1	NMOS	D	16		SOM	YES	3=ST	YES	1	+12=5=5	315		500	350
MK4096P1	MOSTEK	RAM	4K	4K X 1		D	16							+12=5=12	400		400	450
MK4096P6	MOSTEK	RAM	4K	4K X 1		D	16							+12=5=12	400		200	200
MK4096P7	MOSTEK	RAM	4K	4K X 1		D	16							+12=5=12	400		300	300
MK4096-11	MOSTEK	RAM		4K X 1	NMOS	D D					3=ST			+12=5=5					350
MK4096-16	MOSTEK	RAM		4K X 1	NMOS	D O					3=ST			+12=5=5					300
4101	INTEL	RAM	1K	256 X 4		STAT													
4102P	MOSTEK	RAM	1K	1K X 1	NMOS	STAT	16	SEP	YES	YES	3=ST	YES	1	+5		350		1000	1000
MK4102-1	MOSTEK	RAM		1K X 1	NMOS	STAT					3=ST			+5					450
MK4102=2	MOSTEK	RAM		1K X 1	NMOS	STAT					3=ST			+5					275
MK4102N1	MOSTEK	RAM	1K	1K X 1	NMOS	STAT	16	SEP	YES	YES	3=ST	YES	1	+5					450
MK4102N6	MOSTEK	RAM	1K	1K X 1	NMOS	STAT	16	SEP	YES	YES	3=ST	YES	1	+5					275
4102P=1	MOSTEK	RAM	1K	1K X 1	NMOS	STAT	16	SEP	YES	YES	3=ST	YES	1	+5		350		450	450
MK4102P1	MOSTEK	RAM	1K	1K X 1	NMOS	STAT	16	SEP	YES	YES	3=ST	YES	1	+5					450
MK4102P6	MOSTEK	RAM	1K	1K X 1	NMOS	STAT	16	SEP	YES	YES	3=ST	YES	1	+5					275
4104	EMM	RAM		1K X 4	NMOS	STAT					3=ST			+5=12=5					250
UP04-11-3	NIPPON	RAM		4K X 1	NMOS	O O					3=ST			+5=12					150
EA4116	ELEC ARR	RAM		4K X 1	NMOS	D D					3=ST			+5=5=12					200
MK4116P=2	MOSTEK	RAM	16K	16K X 1	NMOS	O	16	MUX	YES	YES			YES	1 +12=5=5	460		375	150
MK4116P=3	MOSTEK	RAM	16K	16K X 1	NMOS	O	16	MUX	YES	YES			YES	1 +12=5=5	460		375	200
EA4122=2	ELEC ARR	RAM		4K X 1	NMOS	O O					3=ST			+5=5=12					200
EA4122=1	ELEC ARR	RAM		4K X 1	NMOS	D D					3=ST			+5=12=5					250
EA4122	ELEC ARR	RAM		4K X 1	NMOS	D D					3=ST			+12=5=5					300
4200		EMM		4K X 1	NMOS	STAT					3=ST			+12=5					200
MK4200-11	MOSTEK	RAM		4K X 1	NMOS	D D					3=ST			+12=5=5					350
MK4200=16	MOSTEK	RAM		4K X 1	NMOS	D D					3=ST			+12=5=5					300
TMS4200	TI	MASK		1K X 4	PMOS						3=ST			+5=12					500
MM4203Q	NAT SEMI	PROM		256 X 8	PMOS						3=ST			+5=12					1000
MM4203Q	NAT SEMI	PROM		512 X 4	PMOS						3=ST			+5=12					1000
MM4203Q	NATIONAL	EPROM		256 X 8	PMOS						3=ST			+5=12					1000
MM4203Q	NAT SEMI	EPROM		512 X 8	PMOS						3=ST			+5=12					1250
MM4204	NAT SEMI	EPROM		512 X 8	PMOS						3=ST			+5=12					1250
MM4210	NAT SEMI	MASK		256 X 4	PMOS									+12=12					650
MM4211	NAT SEMI	MASK		256 X 4	PMOS									+5=12					950
MM4213	NAT SEMI	MASK		256 X 8	PMOS									-12=9					850
MM4213=OK	NAT SEMI	MASK		512 X 4	PMOS						3=ST			-12					850
MM4214	NAT SEMI	MASK		512 X 8	PMOS									+5=12					1000
MM4220	NAT SEMI	MASK		256 X 4	PMOS									+12=12					650
MM4221	NAT SEMI	MASK		256 X 4	PMOS									+5=12					950
MM4229	NAT SEMI	MASK		256 X12	PMOS									+5=12					1400
MM4230	NAT SEMI	MASK		256 X 8	PMOS									+12=12					750
MM4230	NAT SEMI	MASK		512 X 4	PMOS									+12=12					725
MM4231	NAT SEMI	MASK		256 X 8	PMOS									-12=9					950
MM4231	NAT SEMI	MASK		512 X 4	PMOS									+5=12					950
MM4232	NAT SEMI	MASK		512 X 8	PMOS									+5=12					1000
MM4232	NAT SMMI	MASK		1K X 4	PMOS									+5=12					1000
MM4240	NAT SEMI	MASK		512 X 5	PMOS									+12=12					500
MM4241	NAT SEMI	MASK		384 X 8	PMOS						3=ST			+5=12					900
MM4250	NAT SEMI	RAM		256 X 1	PMOS	STAT					3=ST			+5=12					650
MM4261	NAT SEMI	RAM		1K X 1	PMOS	D D					OP C			+5=7-12					450
MCM4303A	MOTOROLA	MASK		64 X 8	TTL									+5					75
MC4304	MOTOROLA	RAM		16 X 1	TTL	STAT					OP C			+5					35
MC4305	MOTOROLA	RAM		16 X 1	TTL	STAT					OP C			+5					35
4308	INTEL	MASK	8K		X														
4316	INTEL	MASK	16K		X														
MCM4364	MOTOROLA	RAM		16 X 4	TTL	STAT					OP C			+5					60
4401	EMM	RAM	4K	4K X 1	NMOS	STAT	22				YES					180			
4402	EMM	RAM	4K	4K X 1	NMOS	STAT	22				3=ST	NO		-		300		300	200

NUMBER	MANUF	TYPE	TOTAL BITS	ORG WDSXBTS	PROC	MODE	PI NS	I/N/ OUT	-INPUTS- DEC.	TTL	OUTPUTS TYPE	TTL	CS	POWER SUPPLIES	PWR MW	SPEED=NANOSEC RMW	RECV	ACC
3515	FAIRCHILD	MASK	4K	512 X 8	PMOS	STAT	24			YES	3=ST	YES	4	-12=5	510			600
3524=5	FAIRCHILD	RAM	1K	1K X 1		D												
3538	FAIRCHILD	RAM	1K	256 X 4	NMOS	D U	22	SEP	YES	YES	3=ST	YES	1	+5	650		650	650
3538=1	FAIRCHILD	RAM	1K	256 X 4	NMOS	D U	22	SEP	YES	YES	3=ST	YES	1	+5	350		450	450
3538=2	FAIRCHILD	RAM		256 X 4	NMOS	STAT					3=ST			+5				650
35L38	FAIRCHILD	RAM	1K	256 X 4	NMOS	D U	22	SEP	YES	YES	3=ST	YES	1	+5	184		650	650
35L38A	FAIRCHILD	RAM	1K	256 X 4	NMOS	D U	22	SEP	YES	YES	3=ST	YES	1	+5	184		500	500
35L38B	FAIRCHILD	RAM	1K	256 X 4	NMOS	D U	22	SEP	YES	YES	3=ST	YES	1	+5	184		400	400
3538F	FAIRCHILD	RAM	1K	256 X 4	NMOS	D U	22	SEP	YES	YES	3=ST	YES	1	+5	350		350	350
3539	FAIRCHILD	RAM	2K	256 X 8	NMOS	D U	22	MUX		YES	3=ST	YES	2	+5	500		650	650
3539=1	FAIRCHILD	RAM	2K	256 X 8	NMOS	D U	22	MUX		YES			2	+5	500		400	400
3539=2	FAIRCHILD	RAM	2K	256 X 8	NMOS	D U	22	MUX		YES			2	+5	500		500	500
3542	FAIRCHILD	RAM		1K X 1	NMOS	STAT					3=ST			+5				150
3542=2	FAIRCHILD	RAM		1K X 1	NMOS	STAT					3=ST			+5				120
D3601	INTEL	PROM	1K	256 X 4	TTL			16		YES	OP C	YES	2	+5	685			70
D3601-1	INTEL	PROM	1K	256 X 4	TTL			16		YES	OP C	YES	2	+5	685			50
M3601	INTEL	PROM		256 X 4	TTL						OP C			+5				90
D3602	INTEL	PROM	2K	512 X 4	TTL						OP C			+5	735			70
D3602=4	INTEL	PROM	2K	512 X 4	TTL						OP C			+5	735			90
3602L=8	INTEL	PROM		512 X 4	TTL						OP C			+5				120
D3604	INTEL	PROM	4K	512 X 8	TTL						OP C			+5	998			70
D3604=4	INTEL	PROM	4K	512 X 8	TTL						OP C			+5	998			90
3604L=8	INTEL	PROM		512 X 8	TTL						3=ST			+5				120
D3621	INTEL	PROM	1K	256 X 4	TTL						3=ST				685			70
D3621-1	INTEL	PROM	1K	256 X 4	TTL						3=ST			+5	685			50
D3622	INTEL	PROM	2K	512 X 4	TTL						3=ST			+5	735			70
D3622=4	INTEL	PROM	2K	512 X 4	TTL						3=ST			+5	735			90
3622L=8	INTEL	PROM		512 X 4	TTL						3=ST			+5				120
D3624	INTEL	PROM			TTL						3=ST			+5	735			70
D3624=4	INTEL	PROM			TTL						3=ST			+5	735			90
3625	INTEL	PROM		1K X 4	TTL						3=ST			+5				70
MK3702	MOSTEK	EPROM		256 X 8	PMOS						3=ST			+5=9				450
MK3702=1	MOSTEK	EPROM	2K	256 X 8			24											550
MK3702-2	MOSTEK	EPROM	2K	256 X 8			24											750
MK3702=3	MOSTEK	EPROM	2K	256 X 8			24											650
EA3800	ELEC ARR	MASK		1K X11	PMOS									-12=24				3600
R03=8316A	EMM	MASK		2K X 8	NMOS						3=ST			+5				850
R03=8316B	EMM	MASK		2K X 8	NMOS						3=ST			+5=12				450
R03=8316A	GEN INST	MASK		2K X 8	NMOS						3=ST			+5				550
R03=8316B	GEN INST	MASK		2K X 8	NMOS						3=ST			+5=12				550
38510/201	HARRIS	PROM		64 X 8	TTL						OP C			+5				75
EA4000	ELEC ARR	MASK		512 X10	PMOS									-12=12				725
CRC4001	COLLINS	RAM		16 X 4	PMOS	STAT								-27=19				400
4001	INTEL	RAM	2K	X														
MCM4002	MOTOROLA	MASK		32 X 8	TTL						OP C			+5				90
4002	INTEL	RAM	320	X														
MCM4003A	MOTOROLA	MASK		64 X 8	TTL									+5				75
MC4004	MOTOROLA	RAM		16 X 1	TTL	STAT					OP C			+5				35
MCM4004A	MOTOROLA	MASK		256 X 4	TTL						OP C			+5				50
MC4005	MOTOROLA	RAM		16 X 1	TTL	STAT					OP C			+5				35
S4006	AMI	RAM	1K	1K X 1	PMOS	O	16				OP C			-12=5				400
MK4006	MOSTEK	RAM	1K	1K X 1	PMOS	O D	16				OP C			-12=9			400	400
MCM4006A	MOTOROLA	MASK		256 X 4	TTL						OP C			-				60
CD4006I	RCA	RAM		256 X 1	CMOS	STAT					3=ST			+10				100
MK4007	MOSTEK	RAM	256	256 X 1	CMOS	STAT	16			YES				+5=12				900
S4008	AMI	RAM		1K X 1	PMOS	D D					OP C			+5=12				500
S4008=9	AMI	RAM		1K X 1	PMOS	D O					OP C			+5=12				800
MK4008P6	MOSTEK	RAM	1K	1K X 1	PMOS	O D	16				OP C			+5=12				900
MK4008P9	MOSTEK	RAM	1K	1K X 1		O	16											900
CD4016A	RCA	RAM		256 X 1	CMOS	STAT					3=ST			3TO15				200
S4021	AMI	RAM		4K X 1	NMOS	O O					3=ST			+12=5=5				400
S4021=2	AMI	RAM		4K X 1	NMOS	O O					3=ST			+12=9=5				300
S4021=3	AMI	RAM		4K X 1	NMOS	O O					3=ST			+5=12=9				250
S4021=4	AMI	RAM		4K X 1	NMOS	O O					3=ST			+5=5=12				200
MK4027	MOSTEK	RAM	4K	4K X 1	NMOS	D O	16				3=ST			+12=9=5				300
TMS4030	TI	RAM	4K	4K X 1	NMOS	D 2	22	SEP	YES	SOME	3=ST	YES	1	12=3=5	400	710	470	300
TMS4030=1	TI	RAM	4K	4K X 1	NMOS	D 2	22	SEP	YES	SOME	3=ST	YES	1	12=3=5	400	640	430	250
TMS4030=2	TI	RAM	4K	4K X 1	NMOS	D 2	22	SEP	YES	SOME	3=ST	YES	1	12=3=5	400	580	400	200
TMS4033	TI	RAM	1K	1K X 1	NMOS	STAT	16	SEP	YES	YES	3=ST	YES	1	+5	350		450	450
TMS4034	TI	RAM	1K	1K X 1	NMOS	STAT	16	SEP	YES	YES	3=ST	YES	1	+5	350		650	650
TMS4035	TI	RAM	1K	1K X 1	NMOS	STAT	16	SEP	YES	YES	3=ST	YES	1	+5	350		1000	1000
CD4036A	RCA	RAM		32 X 8	CMOS	STAT				YES	3=ST			3TO15				1000
CM4036A	SOLITRON	RAM		4 X 8

NUMBER	MANUF	TYPE	TOTAL BITS	ORG #WDS	BITS	PROC	MODE	PI NS	INV OUT	=INPUTS= DEC	OUTPUTS TTL	CS	POWER SUPPLIES	PWR Mw	SPEED=NANOSEC RMw	RdCY	ACC
5275	MMI	MASK		2K	X 8	TTL					OP C		+5				120
5276	MMI	MASK		2K	X 8	TTL					3-ST		+5				120
5280	MMI	MASK		1K	X 8	TTL					OP C		+5				175
MM5280	NAT SEMI	RAM		4K	X 1	NMOS	O	O			3-ST		+12+5=5				200
5281	MMI	MASK		1K	X 8	TTL					3-ST		+5				175
5300	MONO MEM	PROM	1K	256	X 4	TTL		16		YES YES	OP C	YES	2 +5				75
5301	MONO MEM	PROM	1K	256	X 4	TTL		16		YES YES	3-ST	YES	2 +5				75
5301-1	M M I	PROM		256	X 4	TTL					3-ST		+5				60
MCM5303	MOTOROLA	PROM		64	X 8	TTL					OP C		+5				75
5305	MONO MEM	PROM	2K	512	X 4			16			3-ST	YES					70
5305-1	M M I	PROM		512	X 4	TTL					OP C		+5				70
5306	MONO MEM	PROM	2K	512	X 4			16			3-ST	YES					75
5306-1	M M I	PROM		512	X 4	TTL					OP C		+5				70
5330	MONO MEM	PROM	256	32	X 8	TTL		16		YES YES	OP C	YES	1 +5				60
RC5330	RAYTHEON	RAM		256	X 1	TTL	STAT				OP C		+5				55
RM5330	RAYTHEON	RAM		256	X 1	TTL	STAT				OP C		+5				75
5331	MONO MEM	PROM	256	32	X 8	TTL		16		YES YES	3-ST	YES	1 +5				60
5335-1	M M I	PROM		256	X 8	TTL					OP C		+5				90
5336-1	M M I	PROM		256	X 8	TTL					3-ST		+5				90
5340	MONO MEM	PROM	4K	512	X 8	TTL		24		YES YES	OP C	YES	+5	700			90
RC5340	RAYTHEON	RAM		256	X 1	TTL	STAT				3-ST		+5				90
RM5340	RAYTHEON	RAM		256	X 1	TTL	STAT				3-ST		+5				70
5341	MONO MEM	PROM	4K	512	X 8	TTL		24		YES YES	3-ST	YES	+5	700			90
5350	MMI	PROM		1K	X 4	TTL					OP C		+5				75
5351	MMI	PROM		1K	X 4	TTL					3-ST		+5				75
5352	MONO MEM	PROM	4K	1K	X 4			18			OP C	YES					75
5353	MONO MEM	PROM	4K	1K	X 4			18			3-ST	YES					75
5380	MMI	PROM		1K	X 8	TTL					OP C		+5				70
5381	MMI	PROM		1K	X 8	TTL					OP C		+5				70
R5500	RAYTHEON	RAM		1K	X 1	TTL	STAT				OP C		+5				35
I45501	INTERSIL	RAM	64	16	X 4	TTL	STAT	16	SEP	YES YES	OP C	YES	1 +5				60
MM5501	RCA	RAM		1K	X 1	CMOS	STAT				3-ST		+10				80
SCL256.4015	SSS	RAM		256	X 4	CMOS	STAT				3-ST		-				100
MM55502	RCA	RAM		1K	X 1	CMOS	STAT				3-ST		+5				150
SCM55025	SSS	RAM		1K	X 1	CMOS	STAT				3-ST		+3TO+15				100
I45503	INTERSIL	RAM	256	256	X 1	TTL	STAT	16	SEP	YES YES	OP C	YES	3 +5				80
I45503M	INTERSIL	RAM		256	X 1	TTL	STAT				OP C		+5				80
I45503AC	INTERSIL	RAM		256	X 1	TTL	STAT				OP C		+5				60
I45503AM	INTERSIL	RAM		256	X 1	TTL	STAT				OP C		+5				70
I45508C	INTERSIL	RAM		1K	X 1	NMOS	STAT				OP C		+5				100
I45508C	INTERSIL	RAM		1K	X 1	TTL	STAT				OP C		+5				70
I45508M	INTERSIL	RAM		1K	X 1	TTL	STAT				OP C		+5				75
I45518C	INTERSIL	RAM		1K	X 1	TTL	STAT				3-ST		+5				70
I45518M	INTERSIL	RAM		1K	X 1	TTL	STAT				3-ST		+5				75
SCM55205	SSS	RAM		256	X 1	CMOS	STAT				3-ST		+3TO15				100
I45523C	INTERSIL	RAM	256	256	X 1	TTL	STAT	16	SEP	YES YES	3-ST	YES	3 +5				80
I45523M	INTERSIL	RAM		256	X 1	TTL	STAT				3-ST		+5				80
I45523AC	INTERSIL	RAM		256	X 1	TTL	STAT				3-ST		+5				60
I45523AM	INTERSIL	RAM		256	X 1	TTL	STAT				3-ST		+5				70
5530	MONO MEM	RAM	256	256	X 1	TTL	STAT	16	SEP	YES YES	OP C	YES	3 +5				70
L5530	MONO MEM	RAM		256	X 1	TTL	STAT				OP C		+5				130
5531	MONO MEM	RAM	256	256	X 1	TTL	STAT	16	SEP	YES YES	3-ST	YES	3 +5				70
L5531	MMI	RAM		256	X 1	TTL	STAT				3-ST		+5				130
IM5533AC	INTERSIL	RAM		256	X 1	TTL	STAT				OP C		+5				60
IM5533AM	INTERSIL	RAM		256	X 1	TTL	STAT				OP C		+5				70
IM5533C	INTERSIL	RAM		256	X 1	TTL	STAT				OP C		+5				80
IM5533M	INTERSIL	RAM		256	X 1	TTL	STAT				OP C		+5				80
IM5553	INTERSIL	RAM	256	256	X 1	TTL	STAT	16	SEP	YES YES	OP C	YES	3 +5				
MM55540	RCA	RAM		256	X 4	CMOS	STAT				3-ST		+10				100
5555	MMI	RAM		64	X 9	TTL	STAT						+5				80
5560	MONO MEM	RAM	64	16	X 4	TTL	STAT	16	SEP	YES YES	OP C	YES	1 +5				60
L5560	MMI	RAM		16	X 4	TTL	STAT				3-ST		+5				60
5561	MONO MEM	RAM	64	16	X 4	TTL	STAT	16	SEP	YES YES	3-ST	YES	1 +5				100
L5561	MMI	RAM		16	X 4	TTL	STAT				3-ST		+5				60
IM5600	INTERSIL	PROM	256	32	X 8	TTL		16		YES YES	OP C	YES	1 +5				100
IM5600CF	INTERSIL	PROM		32	X 8	TTL					OP C		+5				50
IM5603	INTERSIL	PROM	1K	256	X 4	TTL		16		YES YES	OP C	YES	2 +5				70
IM5604	INTERSIL	PROM		512	X 4	TTL					OP C		+5				60
IM5605	INTERSIL	PROM		512	X 8	TTL					OP C		+5				70
IM5610	INTERSIL	PROM	256	32	X 8	TTL		16		YES YES	3-ST	YES	1 +5				80
IM5610CF	INTERSIL	PROM		32	X 8	TTL					3-ST		+5				50
IM5623	INTERSIL	PROM	1K	256	X 4	TTL		16		YES YES	3-ST	YES	2 +5				70
IM5624	INTERSIL	PROM	2K	512	X 4	TTL					3-ST		+5	500			60
IM5625	INTERSIL	PROM		512	X 8	TTL					3-ST		+5				50
RO5-8192	GEN INST	MASK		2K	X 4	PMOS							+5-12				80
AM56002	AMS	RAM		1K	X 1	PMOS	D	D			OP C		20+22.5+7				1600
6002	NORTEC	RAM		1K	X 1	PMOS	D	D			OP C		20+22.5+7				150
AM5600311	AMS	RAM		2K	X 1	PMOS	D	D			OP C		+5+8-15				150
AM5600310	AMS	RAM		2K	X 1	PMOS	D	D			OP C		+5+8-15				350
600311	NORTEC	RAM		2K	X 1	PMOS	D	D			OP C		+5+8-15				460
6200	MONO MEM	MASK	1K	256	X 4	TTL		16		YES YES	OP C	YES	2 +5	525			350
M6201	MMI	MASK		256	X 4	TTL					OP C		+5				70
6205	MONO MEM	MASK	2K	512	X 4	TTL		16		YES YES	OP C	YES	1 +5				45
A206	MONO MEM	MASK	2K	512	X 4	TTL		16		YES YES	3-ST	YES	1 +5				60
6210	MMI	MASK		256	X 5	TTL					OP C		+5				60
6225	MMI	MASK		512	X 5	TTL					OP C		+5				75

NUMBER	MANUF	TYPE	TOTAL BITS	ORG #WDS	BITS	PROC	MODE	PI NS	INV OUT	=INPUTS= DEC	OUTPUTS TTL	CS	POWER SUPPLIES	PWR Mw	SPEED=NANOSEC RMw	RdCY	ACC
MM5628	HARRIS	RAM		256	X 4	CMOS	STAT				3-ST		+4TO+11				305
MM5628C	HARRIS	RAM		256	X 4	CMOS	STAT				3-ST		+4TO+11				160
MM6562-2	HARRIS	RAM		256	X 4	CMOS	STAT				3-ST		+5				465
MM6562-9	HARRIS	RAM		256	X 4	CMOS	STAT				3-ST		+4TO+11				425
MCM6590	MOTOROLA	MASK		2K	X 8	NMOS					3-ST		+5+12-3				800
NCM6590	NITRON	MASK		2K	X 8	NMOS					3-ST		+5+12-3				800
NCM6590A	NITRON	MASK		2K	X 8	NMOS							+5+5+12				475
MCM6604	MOTOROLA	RAM		4K	X 1	NMOS	D	D			3-ST		+12+5=5				350
MCM6604-2	MOTOROLA	RAM		4K	X 1	NMOS	D	D			3-ST		+5+12=5				250
MCM6604-4	MOTOROLA	RAM		4K	X 1	NMOS	D	D			3-ST		+12+5=5				300
MCM6605	MOTOROLA	RAM	4K	4K	X 1	NMOS	D	2	22	YES		YES	1 +5+12=5	280	470	450	210
MCM6605A	MOTOROLA	RAM		4K	X 1	NMOS	D	D			3-ST		+12+5=5				300
MCM6605A1	MOTOROLA	RAM		4K	X 1	NMOS	D	D			3-ST		+5+5+12				150
MCM6605A2	MOTOROLA	RAM		4K	X 1	NMOS	D	D			3-ST		+12+5=5				200
MCM6606	MOTOROLA	RAM	4K	4K	X 1	NMOS	D	2	22	YES		YES	1 +5+12=5	280			190
S6810	AMI	RAM		128	X 8	NMOS	STAT				3-ST		+5				1000
S6810-1	AMI	RAM		128	X 8	NMOS	STAT				3-ST		+5				575
MCM6810A	MOTOROLA	RAM		128	X 8	NMOS	STAT				3-ST		+5				500
MCM6810AL	MOTOROLA	RAM	1K	128	X 8	NMOS	STAT	24	MUX	YES YES	3-ST	YES	6 +5		350		450
S6830	AMI	MASK		1K	X 1	NMOS					3-ST		+5				575
MCM6830	MOTOROLA	MASK		1K	X 8	NMOS					3-ST		+5				575
MCM6830A	MOTOROLA	MASK		1K	X 8	NMOS					3-ST		+5				500
MCM6830B	MOTOROLA	MASK	8K	1K	X 8	NMOS							+5				
MCM6830L	MOTOROLA	MASK	8K	1K	X 8	NMOS				YES	3-ST		4 +5				800
S6831	AMI	MASK		2K	X 8	NMOS					3-ST		+5				500
MCM6831?	MOTOROLA		16K	2K	X 8	NMOS							+5				
MCM6832	MOTOROLA	MASK		2K	X 8	NMOS					3-ST		+5+5+12				550
NCM6832	NITRON	MASK		2K	X 8	NMOS					3-ST		+5+5+12				550
S6834	AMI	PROM	4K	512	X 8	PMOS			MUX		3-ST		+5+12				575
MCM6832L	MOTOROLA	MASK	16K	2K	X 8	NMOS							+5+12=5				
MCM6870B	MOTOROLA	EAROM	8K	1K	X 8	NMOS							+5+12=5				
7001I	AMS	RAM	1K	1K	X 1	NMOS	STAT			YES			+7+15=3			180	60
MCM7001	MOTOROLA	RAM		1K	X 1	NMOS	STAT						+7+15=3				60
MCM7001-1	MOTOROLA	RAM		1K	X 1	NMOS	STAT						+15+7=3				75
NC7001	NITRON	EAROM		64	X 1	NMOS											800
7001	NORTEC	RAM		1K	X 1	NMOS	STAT						+7+15=3				60
7001	RCA	RAM	1K	1K	X 1	NMOS	D	2		YES			+7+15=3			180	60
NC7002	NITRON	EAROM		512	X 2	NMOS					3-ST		-30				2000
NC7002	NITRON	EAROM		1K	X 1	NMOS					3-ST		-30				2000
AMS7003	AMS	RAM		2K	X 1	NMOS	STAT						+7+15=3				60
AMS7005	AMS	RAM		4K	X 1	NMOS	D	D			3-ST		+5+5+12				200
AMS7008	AMS	RAM		8K	X 1	NMOS	D	D			3-ST		+12+5=5				125
NC7010	NITRON	EAROM		512	X 2	NMOS					3-ST		-30				-
NC7010	NITRON	EAROM		1K	X 1	NMOS					3-ST		-30				-
NC7030	NITRON	EAROM		8	X16	NMOS							+15-15				
NC7030	NITRON	EAROM		16	X 8	NMOS							+15-15				
MM7040	RCA	RAM		256	X 4	NMOS	STAT				3-ST		+5+12=5				200
AMS7101	AMS	RAM		256	X 4	NMOS	STAT				3-ST		+5				250
SY7103	SYNERTEK	RAM		1K	X 1	NMOS	O	D			OP C		+5+12				60
AMS7111	AMS	RAM		256	X 4	NMOS	STAT				3-ST		+5				250
AMS7112	AMS	RAM		256	X 4	NMOS	STAT				3-ST		+5				250
NM7202	HARRIS	RAM		1K	X 1	NMOS	STAT				3-ST		+5				1000
MM7202A	HARRIS	RAM		1K	X 1	NMOS	STAT				3-ST		+5				650
MM7202B	HARRIS	RAM		1K	X 1	NMOS	STAT				3-ST		+5				500
MM7202C-2	HARRIS	RAM		1K	X 1	NMOS	STAT				3-ST		+5				400
MM7202C-5	HARRIS	RAM		1K	X 1	NMOS	STAT				3-ST		+5				350
AMS7270	AMS	RAM		4K	X 1	NMOS	O	D			3-ST		+12=5				200
AMS7271	AMS	RAM		4K	X 1	NMOS	D	D			3-ST		+12=5				200
AMS7280	AMS	RAM		4K	X 1	NMOS	D	D			3-ST		+5+5+12				200
74170	FAIRCHILO	RAM		4	X 4	TTL	STAT				OP C		+5				40
74LS170	FAIRCHILO	RAM		4	X 4	TTL	STAT				OP C		+5				40
74172	TI	RAM		8	X 2	TTL	STAT				3-ST		+5				50
74186	TI	PROM	512	64	X 8	TTL		24		YES YES	OP C	YES	2 +5				55
DM74187	NAT SEMI	MASK		256	X 4	TTL					OP C		+5				60
DM74L187A	NAT SEMI	MASK		256	X 4	TTL					OP C		+5				50
74187	TI	MASK	1K	256	X 4	TTL		16		YES YES	OP C	YES	2 +5	525			70
745188	TI	PROM	256	32	X 8	TTL		16		YES YES	OP C	YES	1 +5	330			25
74188A	TI	PROM	256	32	X 8	TTL		16		YES YES	OP C	YES	1 +5	330			34
745189	TI	RAM	64	16	X 4	TTL	STAT	16	SEP	YES YES	3-ST	YES	1 +5				70
74S189	SIGNETICS	RAM	64	16	X 4	TTL	STAT	16	SEP	YES YES	3-ST	YES	1 +5	375		50	50
MD74C200	HARRIS	RAM		256	X 1	CMOS	STAT				3-ST		3TO15				400
MM74C200	NAT SEMI	RAM		256	X 1	CMOS	STAT				3-ST		3TO15				400
DM74S200	NAT SEMI	RAM		256	X 1	TTL	STAT				3-ST		+5				50
MM74C200	TELEDYNE	RAM		256	X 1	CMOS	STAT				3-ST		3TO15				400
74S200	SIGNETICS	RAM	256	256	X 1	TTL	STAT	16	SEP	YES YES	3-ST	YES	3 +5				
74200	TI	RAM	256	256	X 1	TTL				YES YES		YES					
74S201	SIGNETICS	RAM	256	256	X 1	TTL	STAT	16	SEP	YES YES	3-ST	YES	3 +5				
74S201	TI	RAM	256	256	X 1	TTL	STAT	16	SEP	YES YES	3-ST	YES	3 +5		400	65	42
SN74S201	TI	RAM		256	X 1	TTL	STAT				3-ST		+5				65
DM74S206	NAT SEMI	RAM		256	X 1	TTL	STAT				OP C		+5				60
74S209	TI	RAM	1K	1K	X 1	TTL	STAT	16	SEP	YES YES	3-ST	YES	1 +5		150		70
74S270	TI	MASK	2K	512	X 4	TTL		16		YES YES	OP C	YES	1 +5	600			70
74S271	TI	MASK	2K	256	X 8	TTL		20		YES YES	OP C	YES	2 +5				45
74S287	NAT SEMI	PROM	1K	256	X 4	TTL		16		YES YES	3-ST	YES	2 +5				70
74S287	TI	PROM	1K	256	X 4	TTL		16		YES YES	3-ST	YES	2 +5	525			70
74S288	TI	PROM	256	32	X 8	TTL		16		YES YES	3-ST	YES	1 +5				25

NUMBER	MANUF	TYPE	TOTAL BITS	ORG #WDS	BITS	PROC	MODE	PI NS	INV OUT	=INPUTS= DEC	OUTPUTS TTL	CS	POWER SUPPLIES	PWR Mw	SPEED=NANOSEC RMw	RdCY	ACC
6230	MONO MEM	MASK	256	32	X 8	TTL		16		YES YES	OP C	YES	1 +5				26
6231	MMI	MASK		32	X 8	TTL					3-ST		+5				50
6235	MMI	MASK		256	X 8	TTL					OP C		+5				60
M6240	MMI	MASK		512	X 8	TTL					OP C		+5				175
H6240	MMI	MASK		512	X 8	TTL					OP C		+5				75
A6241	MMI	MASK		512	X 8	TTL					3-ST		+5				175
H6241	MMI	MASK		512	X 8	TTL					3-ST		+5				75
6255	MMI	MASK		1K	X10	TTL					OP C		+5				120
6260	MMI	MASK		1K	X 9	TTL					OP C		+5				150
6275	MMI	MASK		2K	X 8	TTL					OP C		+5				110
6276	MMI	MASK		2K	X 8	TTL					3-ST		+5				110
6280	MMI	MASK		1K	X 8	TTL					OP C		+5				150
6281	MMI	MASK		1K	X 8	TTL					3-ST		+5				150
6300	MONO MEM	PROM	1K	256	X 4	TTL		16		YES YES	OP C	YES	2 +5				55
6301	MONO MEM	PROM	1K	256	X 4	TTL		16		YES YES	3-ST	YES	2 +5				55
6305	MONO MEM	PROM	2K	512	X 4	TTL		16			OP C	YES	+5				60
6306	MONO MEM	PROM	2K	512	X 4			16			OP C	YES					60
6306-1	M M I	PROM		512	X 4	TTL					3-ST		+5				70
IM6312	INTERSIL	MASK	12K	1K	X12	CMOS	STAT	18	MUX	YES YES	3-ST	YES	2 +5				500
IM6312A	INTERSIL	MASK	12K	1K	X12	CMOS	STAT	18	MUX	YES NO	3-ST	NO	2 +4TO+11				500
MM6312	HARRIS	MASK		1K	X12	CMOS					3-ST		+5				500
MM6312A	HARRIS	MASK		1K	X12	CMOS					3-ST		+4TO+11				250
6330	MONO MEM	PROM	256	32	X 8	TTL		16		YES YES	OP C	YES	1 +5				50
6331	MONO MEM	PROM	256	32	X 8	TTL		16		YES YES	3-ST	YES	1 +5				50
6335-1	M M I	PROM		256	X 8	TTL					OP C		+5				90
6336-1	M M I	PROM		256	X 8	TTL					3-ST		+5				90
6340	MONO MEM	PROM	4K	512	X 8			24			OP C	YES	+5				90
6341	MONO MEM	PROM	4K	512	X 8	TTL		24			3-ST	YES	+5				90
6350	MMI	PROM		1K	X 4	TTL					OP C		+5				60
6351	MMI	PROM		1K	X 4	TTL					3-ST		+5				60
6352	MONO MEM	PROM	4K	1K	X 4			18			OP C	YES					60
6353	MONO MEM	PROM	4K	1K	X 4			18			3-ST	YES					60
6380	MMI	PROM		1K	X 8	TTL					OP C		+5				70
6381	MMI	PROM		1K	X 8	TTL					OP C		+5				70
MM6508A	HARRIS	RAM		1K	X 1	CMOS	STAT				3-ST		+4TO+11				100
MM6508B	HARRIS	RAM		1K	X 1	CMOS	STAT				3-ST		+5				305
MM6508C	HARRIS	RAM		1K	X 1	CMOS	STAT				3-ST		+4TO+11				160
MM6508-9	HARRIS	RAM		1K	X 1	CMOS	STAT				3-ST		+5				425
MM6508-2	HARRIS	RAM		1K	X 1	CMOS	STAT				3-ST		+5				465
IM6508	INTERSIL	RAM	1K	1K	X 1	CMOS	STAT	16	SEP	YES YES	3-ST	YES	1 +5				460
IM6508-1	INTERSIL	RAM	1K	1K	X 1	CMOS	STAT	16	SEP	YES YES	3-ST	YES	1 +5				300
IM6508A	INTERSIL	RAM	1K	1K	X 1	CMOS	STAT	16	SEP	YES NO	3-ST	NO	2 +4TO+11				350
IM6508A-1	INTERSIL	RAM	1K	1K	X 1	CMOS	STAT	16	SEP	YES NO	3-ST	NO	2 +4TO+11				200
MM6518A	HARRIS	RAM		1K	X 1	CMOS	STAT				3-ST		+4TO+11				100
MM6518B	HARRIS	RAM		1K	X 1	CMOS	STAT				3-ST		+5				305
MM6518C	HARRIS	RAM		1K	X 1	CMOS	STAT				3-ST		+4TO+11				160
MM6518-2	HARRIS	RAM		1K	X 1	CMOS	STAT				3-ST		+5				465
MM6518-9	HARRIS	RAM		1K	X 1	CMOS	STAT				3-ST		+5				425
IM6518	INTERSIL	RAM	1K	1K	X 1	CMOS	STAT	18	SEP	YES YES	3-ST	YES	3 +5				460
IM6518-1	INTERSIL	RAM	1K	1K	X 1	CMOS	STAT	18	SEP	YES YES	3-ST	YES	3 +5				300
IM6518A	INTERSIL	RAM	1K	1K	X 1	CMOS	STAT	18	SEP	YES NO	3-ST	NO	3 +4TO+11				350
IM6518A-1	INTERSIL	RAM	1K	1K	X 1	CMOS	STAT	18	SEP	YES NO	3-ST	NO	3 +4TO+11				200
IM6523	INTERSIL	RAM	256	256	X 1	CMOS					3-ST		3 +5			10	400
IM6524	INTERSIL	RAM	256	256	X 1	CMOS	STAT	16	SEP	YES YES	3-ST	YES	2 +5				585 415
IM6524-1	INTERSIL	RAM	256	256	X 1	CMOS	STAT	16	SEP	YES YES	3-ST	YES	2 +5				410 315
IM6524A	INTERSIL	RAM	256	256	X 1	CMOS	STAT	16	SEP	YES NO	3-ST	NO	2 +4TO+11				410 320
IM6524A-1	INTERSIL	RAM	256	256	X 1	CMOS	STAT	16	SEP	YES NO	3-ST	NO	2 +4TO+11				280 210
6530	MONO MEM	RAM	256	256	X 1	TTL	STAT	16	SEP	YES YES	OP C	YES	3 +5				55
L6530	MMI	RAM		256	X 1	TTL	STAT				OP C		+5				115
6531	MONO MEM	RAM	256	256	X 1	TTL	STAT	16	SEP	YES YES	3-ST	YES	3 +5				55
MM6551A	HARRIS	RAM		256	X 4	CMOS	STAT				3-ST		+4TO+11				100
MM6551B	HARRIS	RAM		256	X 4	CMOS	STAT				3-ST		+4TO+11				305
MM6551C	HARRIS	RAM		256	X 4	CMOS	STAT				3-ST		+4TO+11				160
MM6551-2	HARRIS	RAM		256	X 4	CMOS	STAT				3-ST		+5				465
MM6551-9	HARRIS	RAM		256	X 4	CMOS	STAT				3-ST		+4TO+11				425
IM6551	INTERSIL	RAM	1K	256	X 4	CMOS	STAT	22	SEP	YES YES	3-ST	YES	1 +5				460
IM6551A	INTERSIL	RAM	1K	256	X 4	CMOS	STAT	22	SEP	YES NO	3-ST	NO	1 +4TO+11				350
6555	MMI	RAM		64	X 9	TTL	STAT				OP C		+5				80
6560	MONO MEM	RAM	64	16	X 4	TTL	STAT	16	SEP	YES YES	OP C	YES	1 +5				35
L6560	MMI	RAM		16	X 4	TTL	STAT				OP C		+5				80
MCM6560	MOTOROLA	MASK		1K	X 8	NMOS					3-ST		+5+12=3				350
MCM6560	MOTOROLA	MASK		2K	X 4	NMOS					3-ST		+12+5=3				350
NCM6560	NITRON	MASK		1K	X 8	NMOS					3-ST		+5+12=3				350
NCM6560	NITRON	MASK		2K	X 4	NMOS					3-ST		+12+5=3				350
NCM6560A	NITRON	MASK		1K	X 8	NMOS							+5+5+12				290
NCM6560A	NITRON	MASK		2K	X 4	NMOS							+5+12=5				290
6561	MONO MEM	RAM	64	16	X 4	TTL	STAT	16	SEP	YES YES	3-ST	YES	1 +5				35
L6561	MMI	RAM		16	X 4	TTL	STAT				3-ST		+5				80
MM6561A	HARRIS	RAM		256	X 4	CMOS	STAT				3-ST		+4TO+11				100
MM6561B	HARRIS	RAM		256	X 4	CMOS	STAT				3-ST		+4TO+11				305
MM6561C	HARRIS	RAM		256	X 4	CMOS	STAT				3-ST		+4TO+11				160
MM6561-2	HARRIS	RAM		256	X 4	CMOS	STAT				3-ST		+5				465
MM6561-9	HARRIS	RAM		256	X 4	CMOS	STAT				3-ST		+4TO+11				425
IM6561	INTERSIL

NUMBER	MANUF	TYPE	TOTAL BITS	ORG WDSXBTS	PROC	MODE	PI NS	IN/ OUT	-INPUTS- DEC.*TTL	OUTPUTS TYPE	TTL	CS	POWER SUPPLIES	PWR Mw	SPEED=NANOSEC RMw	RWCY	ACC
82S10	SIGNETICS	RAM	1K	1K X 1	TTL	STAT	16	SEP	YES YES	OP C	YES	1	+5				45
82S11	SIGNETICS	RAM	1K	1K X 1	TTL	STAT	16	SEP	YES YES	3-ST	YES	1	+5				45
82S110	SIGNETICS	RAM	1K	1K X 1	TTL	STAT	16	SEP	YES YES	OP C	YES	1	+5				35
82S111	SIGNETICS	RAM	1K	1K X 1	TTL	STAT	16	SEP	YES YES	3-ST	YES	1	+5				35
82S112	SIGNETICS	RAM	32	8 X 4	TTL	STAT	24		YES YES	3-ST	YES	2	+5	550			30
82S114	SIGNETICS	PROM	2K	256 X 8	TTL		24		YES YES	3-ST	YES	2	+5	650			60
82S114	SIGNETICS	PROM		256 X 8	TTL					3-ST			+5				90
82S115	SIGNETICS	PROM	4K	512 X 8	TTL		24		YES YES	3-ST	YES	2	+5				60
82S115	SIGNETICS	PROM		512 X 8	TTL					3-ST			+5				90
82S116	SIGNETICS	RAM	256	256 X 1	TTL	STAT	16	SEP	YES YES	3-ST	YES	3	+5				40
82S117	SIGNETICS	RAM	256	256 X 1	TTL	STAT	16	SEP	YES YES	OP C	YES	3	+5				40
82S12	SIGNETICS	RAM	32	8 X 4	TTL	STAT	24		YES YES	OP C	YES	2	+5	550			30
82S123	SIGNETICS	PROM	256	32 X 8	TTL		16		YES YES	3-ST	YES	1	+5				65
82S124	SIGNETICS	PROM	8K	2K X 4	TTL		18		YES YES		YES		+5	400			150
82S126	SIGNETICS	PROM	1K	256 X 4	TTL		16		YES YES	OP C	YES	2	+5				
82S129	SIGNETICS	PROM	1K	256 X 4	TTL		16		YES YES	3-ST	YES	2	+5				
82S130	SIGNETICS	PROM	2K	512 X 4	TTL		16		YES YES		YES	1	+5	600			70
82S131	SIGNETICS	PROM	2K	512 X 4	TTL		16		YES YES		YES	1	+5	600			50
82S136	SIGNETICS	PROM	4K	1K X 4	TTL		18		YES YES		YES	2	+5	600			80
82S137	SIGNETICS	PROM	4K	1K X 4	TTL		18		YES YES		YES	2	+5	600			80
82S16	SIGNETICS	RAM	256	256 X 1	TTL	STAT	16	SEP	YES YES	3-ST	YES	3	+5				50
82S17	SIGNETICS	RAM	256	256 X 1	TTL	STAT	16	SEP	YES YES	OP C	YES	3	+5				
82S185	SIGNETICS	PROM	8K	2K X 4	TTL		18		YES YES		YES		+5	400			150
82S21	SIGNETICS	RAM	64	32 X 2	TTL	STAT	16		YES YES		YES	1	+5	500			50
82S214	SIGNETICS	MASK	2K	256 X 8	TTL		24		YES	3-ST	YES	2	+5	650			90
82S215	SIGNETICS	MASK	4K	512 X 8	TTL		24		YES	3-ST	YES	2	+5	650			90
82S226	SIGNETICS	MASK	1K	256 X 4	TTL		16		YES YES	OP C	YES	2	+5	525			50
82S229	SIGNETICS	MASK		256 X 4	TTL					3-ST			+5				50
8223	SIGNETICS	PROM	256	32 X 8	TTL		16		YES YES	OP C	YES	1	+5				
8253	SIGNETICS	PROM	256	32 X 8	TTL		16		YES YES	OP C	YES	1	+5				65
82S230	SIGNETICS	MASK	2K	512 X 4	TTL		16		YES YES	OP C	YES	1	+5				70
82S231	SIGNETICS	MASK	2K	512 X 4	TTL		16		YES YES	3-ST	YES	1	+5				70
82S25	SIGNETICS	RAM	64	16 X 4	TTL	STAT	16		YES		YES	1	+5	400			60
82S27	SIGNETICS	PROM	1K	256 X 4	TTL		16		YES YES		YES	2	+5	600			40
8228	SIGNETICS	MASK	4K	1K X 4	TTL		16		YES YES		YES	0	+5				70
82S280	SIGNETICS	MASK	8K	1K X 8	TTL		24		YES YES	OP C	YES	4	+5	750			125
82S281	SIGNETICS	MASK	8K	1K X 8	TTL		24		YES YES	3-ST	YES	4	+5	750			125
MPROM8256	HARRIS	PROM	256	32 X 8	TTL		16		YES YES	OP C	YES	1	+5				50
8302	INTEL	MASK	2K	512 X 8			24										1000
EA8302BA	ELEC ARR	MASK		1K X 8	NMOS					3-ST			+5				400
MB8308	FUJITSU	MASK	8K	1K X 8			24										
8308	INTEL	MASK	8K	1K X 8			24										450
8316A	INTEL	MASK	16K	2K X 8			24										850
MB8518E	FUJITSU	EPROM	8K	1K X 8			24						+12+5-5	850			650
MB8518H	FUJITSU	EPROM	8K	1K X 8			24						+12+5-5	850			450
DM8531	NAT SEMI	MASK	16K	2K X 8	TTL		24			3-ST							300
DM8573	NAT SEMI	PROM	1K	256 X 4	TTL		16		YES YES	OP C	YES	2	+5				80
DM8574	NAT SEMI	PROM	1K	256 X 4	TTL		16		YES YES	3-ST	YES	2	+5				80
DM8577	NAT SEMI	PROM	256	32 X 8	TTL		16		YES YES	OP C	YES	1	+5				50
DM8578	NAT SEMI	PROM	256	32 X 8	TTL		16		YES YES	3-ST	YES	1	+5				50
DM8581	NAT SEMI	MASK	16K	1K X16	TTL		40										
DM8582	NAT SEMI	RAM	256	256 X 1	TTL	STAT	16	SEP	YES YES	OP C	YES	3	+5				50
DM8596	NAT SEMI	MASK		512 X 8	TTL					3-ST			+5				150
DM8597	NAT SEMI	MASK	2K	512 X 4	TTL		16		YES YES	3-ST	YES	1	+5				60
DM8598	NAT SEMI	MASK		32 X 8	TTL					3-ST			+5				50
DM8599	NAT SEM	RAM	64	16 X 4	TTL	STAT	16	SEP	YES YES	3-ST	YES	1	+5				45
DM86521	NAT SEMI	RAM		32 X 2	TTL	STAT				OP C			+5				50
DM86L97	NAT SEMI	MASK		256 X 4	TTL					3-ST			+5				130
DM86L99	NAT SEMI	RAM		16 X 4	TTL	STAT				3-ST			+5				150
8702A	AMD	EPROM		256 X 8	PMOS					3-ST			+5-9				1300
8702A	INTEL	EPROM	2K	X 8	PMOS		24			3-ST			+5+9				1300
8702A-4	INTEL	EPROM		256 X 8	PMOS					3-ST			+5-9				2300
MM8702A	NATIONAL	EPROM		256 X 8	PMOS					3-ST			+5-9				1300
MM8702A-4	NAT SEMI	EPROM		256 X 8	PMOS					3-ST			+5+9				2300
8708	INTEL	EPROM	8K	1K X 8			24										450
S8771	AMI	MASK		512 X10	PMOS					3-ST			+5-12				450
DM8795	NAT SEMI	MASK		512 X 8	TTL					OP C			+5				150
DM8796	NAT SEMI	MASK		512 X 8	TTL					3-ST			+5				150
S8865	AMI			2K X 4	PMOS					3-ST			+5-12				1300
S8996	AMI	MASK		4K X 4	PMOS					3-ST			+5-12				1800
AM9050C	AMD	RAM	4K	4K X 1		D	18		YES SOME	3-ST	YES		+12-5	750	300		
AM9050D	AMD	RAM	4K	4K X 1		D	18		SOME	3-ST	YES		+12-5	750	250		
AM9050E	AMD	RAM	4K	4K X 1		D	18		SOME	3-ST	YES		+12-5	750	200		
AM9060C	AMD	RAM	4K	4K X 1	NMOS	D	22		YES SOME	3-ST	YES		+12+5-5	750	300	300	
AM9060D	AMD	RAM	4K	4K X 1	NMOS	D	22		SOME	3-ST	YES		+5+12-5	750	250	250	
AM9101A	AMD	RAM	1K	256 X 4	NMOS	STAT	22			3-ST			+5		500	500	
AM9101B	AMD	RAM	1K	256 X 4	NMOS	STAT	22			3-ST			+5		400	400	
AM9101C	AMD	RAM	1K	256 X 4	NMOS	STAT	22			3-ST			+5		300	300	
AM9101D	AMD	RAM	1K	256 X 4	NMOS	STAT	22			3-ST			+5		250	250	
AM91L01A	AMD	RAM	1K	256 X 4	NMOS	STAT	22			3-ST			+5		500	500	
AM91L01B	AMD	RAM	1K	256 X 4	NMOS	STAT	22			3-ST			+5		400	400	
AM91L01C	AMD	RAM	1K	256 X 4	NMOS	STAT	22			3-ST			+5		300	300	
AM9102A	AMD	RAM	1K	1K X 1	NMOS	STAT	16	SEP	YES YES	3-ST	YES	1	+5		500	500	
AM9102B	AMD	RAM	1K	1K X 1	NMOS	STAT	16	SEP	YES YES	3-ST	YES	1	+5		400	400	
AM9102C	AMD	RAM	1K	1K X 1	NMOS	STAT	16	SEP	YES YES	3-ST	YES	1	+5		300	300	
AM9102D	AMD	RAM	1K	1K X 1	NMOS	STAT	16	SEP	YES YES	3-ST	YES	1	+5		250	250	

NUMBER	MANUF	TYPE	TOTAL BITS	ORG WDSXBTS	PROC	MODE	PI NS	IN/ OUT	-INPUTS- DEC.*TTL	OUTPUTS TYPE	TTL	CS	POWER SUPPLIES	PWR Mw	SPEED=NANOSEC RMw	RWCY	ACC
AM91L02A	AMD	RAM	1K	1K X 1	NMOS	STAT	16	SEP	YES YES	3-ST	YES	1	+5			500	500
AM91L02B	AMD	RAM	1K	1K X 1	NMOS	STAT	16	SEP	YES YES	3-ST	YES	1	+5			400	400
AM91L02C	AMD	RAM	1K	1K X 1	NMOS	STAT	16	SEP	YES YES	3-ST	YES	1	+5			300	300
9107	AMD	RAM		4K X 1	NMOS	D	D			3-ST			+5-5+12				200
9107-4	AMD	RAM		4K X 1	NMOS	D	D			3-ST			+12-5				270
9107-6	AMD	RAM		4K X 1	NMOS	D	D			3-ST			+5+12-5				350
RA-9-1103	GEN INST	RAM	1K	1K X 1	PMOS	D	2	18	SEP YES	OP C	NO	1	19+16				
AM9111A	AMD	RAM	1K	256 X 4	NMOS	STAT	18			3-ST			+5		500	500	
AM9111B	AMD	RAM	1K	256 X 4	NMOS	STAT	18			3-ST			+5		400	400	
AM9111C	AMD	RAM	1K	256 X 4	NMOS	STAT	18			3-ST			+5		300	300	
AM9111D	AMD	RAM	1K	256 X 4	NMOS	STAT	18			3-ST			+5		250	250	
AM91L11A	AMD	RAM	1K	256 X 4	NMOS	STAT	18			3-ST			+5		500	500	
AM91L11B	AMD	RAM	1K	256 X 4	NMOS	STAT	18			3-ST			+5		400	400	
AM91L11C	AMD	RAM	1K	256 X 4	NMOS	STAT	18			3-ST			+5		300	300	
AM9112A	AMD	RAM	1K	256 X 4	NMOS	STAT	16			3-ST			+5		500	500	
AM9112B	AMD	RAM	1K	256 X 4	NMOS	STAT	16			3-ST			+5		400	400	
AM9112C	AMD	RAM	1K	256 X 4	NMOS	STAT	16			3-ST			+5		300	300	
AM9112D	AMD	RAM	1K	256 X 4	NMOS	STAT	16			3-ST			+5		250	250	
AM91L12A	AMD	RAM	1K	256 X 4	NMOS	STAT	16			3-ST			+5		500	500	
AM91L12B	AMD	RAM	1K	256 X 4	NMOS	STAT	16			3-ST			+5		400	400	
AM91L12C	AMD	RAM	1K	256 X 4	NMOS	STAT	16			3-ST			+5		300	300	
AM9130A	AMD		4K	1K X 4		STAT	22		YES		YES		+5	578	500		
AM9130B	AMD		4K	1K X 4		STAT	22								400		
AM9130C	AMD		4K	1K X 4		STAT	22								300		
AM9130D	AMD		4K	1K X 4		STAT	22								250		
AM9130E	AMD		4K	1K X 4		STAT	22								200		
AM9140A	AMD	RAM	4K	4K X 1		STAT	22		YES		YES		+5	578	500		
AM9140B	AMD	RAM	4K	4K X 1		STAT	22								400		
AM9140C	AMD	RAM	4K	4K X 1		STAT	22								300		
AM9140D	AMD	RAM	4K	4K X 1		STAT	22								250		
AM9140E	AMD	RAM	4K	4K X 1		STAT	22								200		
AM9208B	AMD	MASK	8K	1K X 8	NMOS					3-ST			+5+12		400		
AM9208C	AMD	MASK	8K	1K X 8	NMOS					3-ST			+5+12		300		
AM9208D	AMD	MASK	8K	1K X 8	NMOS					3-ST			+5+12		250		
9216	AMD	MASK		512 X 8	NMOS					3-ST			+5			500	
AM9216B	AMD	MASK	16K	2K X 8	NMOS					3-ST			+5+12		400		
AM9216C	AMD	MASK	16K	2K X 8	NMOS					3-ST			+5+12		300		
93404	FAIRCHILD	RAM		16 X 4	TTL	STAT				OP C			+5				35
93405	FAIRCHILD	RAM		16 X 4	TTL	STAT				3-ST			+5				35
F93406	FAIRCHILD	MASK	1K	256 X 4	TTL		16		YES YES	OP C	YES	2	+5	525			70
S93410A	SIGNETICS	RAM	256	256 X 1	TTL	STAT	16	SEP	YES YES	OP C	YES	3	+5				65
93410AC	FAIRCHILD	RAM	256	256 X 1	TTL	STAT	16	SEP	YES YES	OP C	YES	3	+5				45
93410C	FAIRCHILD	RAM	256	256 X 1	TTL	STAT	16	SEP	YES YES	3-ST	YES	3	+5				60
93410M	FAIRCHILD	RAM	256	256 X 1	TTL	STAT	16	SEP	YES YES	OP C	YES	3	+5				70
93411AC	FAIRCHILD	RAM	256	256 X 1	TTL	STAT	16	SEP	YES YES	3-ST	YES	3	+5				45
93411C	FAIRCHILD	RAM	256	256 X 1	TTL	STAT	16	SEP	YES YES	3-ST	YES	3	+5				55
93411M	FAIRCHILD	RAM	256	256 X 1	TTL	STAT	16	SEP	YES YES	3-ST	YES	3	+5				65
93412	FAIRCHILD	RAM		256 X 4	TTL	STAT				OP C			+5				30
93415A	AMD	RAM		1K X 1	TTL	STAT				OP C			+5				45
93415C	AMD	RAM		1K X 1	TTL	STAT				OP C			+5				70
F93415	FAIRCHILD	RAM	1K	1K X 1	TTL	STAT	16	SEP	YES YES	OP C	YES	1	+5				
93415AC	FAIRCHILD	RAM	1K	1K X 1	TTL	STAT	16	SEP	YES YES	OP C	YES	1	+5				45
93415M	FAIRCHILD	RAM	1K	1K X 1	TTL	STAT	16	SEP	YES YES	OP C	YES	1	+5				75
93L415C	FAIRCHILD	RAM	1K	1K X 1	TTL	STAT	16	SEP	YES YES	OP C	YES	1	+5				95
93L415M	FAIRCHILD	RAM	1K	1K X 1	TTL	STAT	16	SEP	YES YES	OP C	YES	1	+5				140
93415C	FAIRCHILD	RAM	1K	1K X 1	TTL	STAT	16	SEP	YES YES	OP C	YES	1	+5				70
93415	RAYTHEON	RAM	1K	1K X 1	TTL	STAT	16	SEP	YES YES	OP C	YES	1	+5				
93415A	SIGNETICS	RAM		1K X 1	TTL	STAT				OP C			+5				45
93416	FAIRCHILD	PROM	1K	256 X 4	TTL		16		YES YES	OP C	YES	2	+5				
93417C	FAIRCHILD	PROM		256 X 4	TTL					OP C			+5				45
93417M	FAIRCHILD	PROM		256 X 4	TTL					OP C			+5				60
93419C	FAIRCHILD	RAM		64 X 9	TTL	STAT				OP C			+5				45
93419M	FAIRCHILD	RAM		64 X 9	TTL	STAT				OP C			+5				60
93L420C	FAIRCHILD	RAM		256 X 1	TTL	STAT				3-ST			+5				45
93L420M	FAIRCHILD	RAM		256 X 1	TTL	STAT				3-ST			+5				55
93421AC	FAIRCHILD	RAM	256	256 X 1	TTL	STAT	16	SEP	YES YES	3-ST	YES	3	+5				40
93421M	FAIRCHILD	RAM	256	256 X 1	TTL	STAT	16	SEP	YES YES	3-ST	YES	3	+5				60
93421C	FAIRCHILD	RAM	256	256 X 1	TTL	STAT	16	SEP	YES YES	3-ST	YES	3	+5				50
93L421C	FAIRCHILD	RAM	256	256 X 1	TTL	STAT	16	SEP	YES YES	3-ST	YES	3	+5				90
93L421M	FAIRCHILD	RAM	256	256 X 1	TTL	STAT	16	SEP	YES YES	3-ST	YES	3	+5				100
93422	FAIRCHILD	RAM		256 X 4	TTL	STAT				3-ST			+5				30
93425AC	FAIRCHILD	RAM	1K	1K X 1	TTL	STAT	16	SEP	YES YES	3-ST	YES	1	+5				45
93425C	FAIRCHILD	RAM	1K	1K X 1	TTL	STAT	16	SEP	YES YES	3-ST	YES	1	+5				70
93425M	FAIRCHILD	RAM	1K	1K X 1	TTL	STAT	16	SEP	YES YES	3-ST	YES	1	+5				75
93L425C	FAIRCHILD	RAM	1K	1K X 1	TTL	STAT	16	SEP	YES YES	3-ST	YES	1	+5				95
93L425M	FAIRCHILD	RAM	1K	1K X 1	TTL	STAT	16	SEP	YES YES	3-ST	YES	1	+5				140
93425A	SIGNETICS	RAM	1K	1K X 1	TTL	STAT	16	SEP	YES YES	3-ST	YES	1	+5				45
F93426	FAIRCHILD	PROM	1K	256 X 4	TTL		16		YES YES	3-ST	YES	2	+5				
93427C	FAIRCHILD	PROM		256 X 4	TTL					3-ST			+5				45
93427M	FAIRCHILD	PROM		256 X 4	TTL					3-ST			+5				60
93431C	FAIRCHILD	MASK		512 X 4	TTL					OP C			+5				50
93431M	FAIRCHILD	MASK		512 X 4	TTL					OP C			+5				60
93432C	FAIRCHILD	MASK		512 X 8	TTL					OP C			+5				55
F93434	FAIRCHILD	MASK	256	32 X 8	TTL		16		YES YES	OP C	YES	1	+5				26
93436	FAIRCHILD	MASK	2K	512 X 4	TTL		16		YES YES	OP C	YES	1	+5				60
93436C	FAIRCHILD	MASK	2K	512 X 4	TTL		16		YES YES	OP C	YES	1	+5				50

NUMBER	MANUF	TYPE	TOTAL BITS	ORG WDSXBTS	PROC	MODE	PI NS	IN/ OUT	-INPUTS- DEC.*TTL	OUTPUTS TYPE	TTL	CS	POWER SUPPLIES	PWR Mw	SPEED=NANOSEC RMw	RWCY	ACC
93438C	FAIRCHILD	PROM		512 X 8	TTL					OP C			+5				55
93438M	FAIRCHILD	PROM		512 X 8	TTL					OP C			+5				70
93441C	FAIRCHILD	MASK		512 X 4	TTL					3-ST			+5				50
93441M	FAIRCHILD	MASK		512 X 4	TTL					3-ST			+5				60
93442C	FAIRCHILD	MASK		512 X 8	TTL					3-ST			+5				55
93446C	FAIRCHILD	PROM		512 X 4	TTL					3-ST			+5				50
93448C	FAIRCHILD	PROM		512 X 8	TTL					3-ST			+5				55
93448M	FAIRCHILD	PROM		512 X 8	TTL					3-ST			+5				70
93456C	FAIRCHILD	MASK		1K X 8	TTL					OP C			+5				60
93457C	FAIRCHILD	MASK		256 X 4	TTL					OP C			+5				45
93457M	FAIRCHILD	MASK		256 X 4	TTL					OP C			+5				60
93464C	FAIRCHILD	MASK		1K X 8	TTL					3-ST			+5				60
93467C	FAIRCHILD	MASK		256 X 4	TTL					3-ST			+5				45
93467M	FAIRCHILD	MASK		256 X 4	TTL					3-ST			+5				60
9610C	FAIRCHILD	RAM		16 X 4	TTL	STAT				3-ST			+5				19
F95400C	FAIRCHILD	RAM		16 X 4	ECL	STAT				OP C			-5.2				25
F95410C	FAIRCHILD	RAM		256 X 1	ECL	STAT							-5.2				40
F95415C	FAIRCHILD	RAM		1K X 1	ECL	STAT							-5.2				75
9702	AMD	EPROM		256 X 8	PMOS					3-ST			+5-9				1000
9702-1	AMD	EPROM		256 X 8	PMOS					3-ST			+5-9				750
S9996	AMI	MASK		2K X 8	PMOS					3-ST			+5-12				1800

10 Print "Editor's Remarks"
30 End
Run

from page 4

masthead of that first issue who stayed with 73, Inc., and are now in *Kilobaud*'s masthead. Funny thing about the mastheads of the first issues of computer magazines Wayne Green starts . . . he was left out in the first issue of *Byte* and he was in the wrong place in the first issue of *Kilobaud*.

Another notable feature regarding the Peterborough crew is that there are a number of women on the staff, and believe me, they're the best-looking in the business . . . and sharp! (As a matter of fact, I've been trying to convince Wayne that I really could use some help out here in California!)

LOOKING BACK APRIL, 1975

Aha, I missed a good one! A great historian I may not be after all! It was *last month*, on March 19th, 1975, that the first meeting of our local computer club here in Lompoc, California took place. Now, this may not seem too significant but my dusty archives indicate that at the time there were no more than three or four computer clubs started in the entire country. The Homebrew Computer Club up in San Francisco was a month old, the Chicago group was a few months old, and the Digital Group was the center of activity in the Denver area. If there were any before these four, I'm sure I'll be hearing about it in my mail!

Hal Singer had just completed an admirable attempt through the *Micro-8 Newsletter* to achieve some standardization within the computer hobbyist community. He received some very good input from some of the top names in the field at the time but it was obvious after seeing them that everyone was going his own way. Hal commented in an editorial summary of the situation, "Everyone's development work has headed in a different direction and our only hope now is to hop onto someone's bandwagon who has done important development work and use his I/O configuration." If only he knew . . .

MISCELLANEOUS

In this month's issue we have a book review by Dave Winthrop on a particular microcomputer dictionary that I personally feel has been long overdue for a raking over the coals. I'd be very interested in getting feedback from you regarding your feelings about the review *and the book*.

Three-State Logic

. . . explanation of a key microprocessor element

John Molnar
Box 561
Ridgefield NJ 07657

Most computer enthusiasts are familiar with the concept of the microcomputer *bus*, those multiwire paths that allow information such as addresses and data to flow between the devices comprising the system. However, without a TTL building block known as Three-state or Tri-state* logic, the microcomputer bus as we know it would be difficult to design and implement in home systems. This article examines Three-state logic as compared to good old regular TTL and open-collector TTL. Three-state logic has an important place in microprocessor memory systems as well as in peripheral and front panel controllers. Hopefully an understanding of Three-state logic will encourage those who were hesitant to design a micro bus system to go ahead and try. However, before jumping directly into a discussion of logic families, a look at the microprocessor bus concept will reveal the *need* for a special type of logic.

*Registered trademark of National Semiconductor.

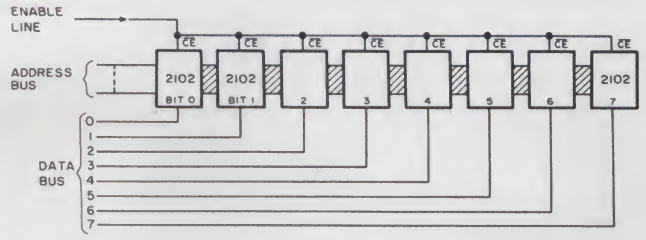


Fig. 1. Simplified 1K memory system using 2102 RAMs. The address bus is common to each chip, as for a given address each 2102 contributes a single bit to the data bus. The enable line, when low, allows the 2102s to be electrically "connected" to the bus. If the enable line is high, each memory chip assumes the open-collector "off" state and is isolated from the data bus.

The Microprocessor Bus

When reading from and writing to memory, most micros specify an address referencing a byte of data somewhere in main memory. The address is 16 bits long in micros such as the 6800 and 8080, and the data is eight bits in length. Address and data information in a micro system are transferred in *parallel*, that is, all bits at once over separate lines. These lines are referred to as a *bus*. Most micros have separate data and address buses, although some use a common bus for all information transfer. Contrast the bus to the single pair of wires connecting a Teletype** to your system. The Teletype uses serial data transmission, that is, one bit at a time is transmitted until an entire character is formed. This one wire bus is fine for peripherals where transmission speed is not critical, but can you imagine how slow your micro would be if all internal data was transferred a bit at a time? An abacus would be faster! Hopefully it can be seen that a parallel bus concept is necessary in order for microprocessor hardware to function efficiently. Let us now take a look at a typical device found on the end of memory bus, the memory chip. Most experimenters are familiar with the 2102 memory, capable of storing 1024 bits. However, it takes eight bits to form a byte, and all eight bits must flow on the bus at the same time. Thus, eight 2102s, each contributing a single bit, are required to form a 1K memory system. Refer to Fig. 1 for a simplified example of a 1K system. However, as most of us know, it takes several kilobytes to run BASIC and many of the applications developed on hobby systems. Hopefully it can be seen that some method of connecting multiple 1K memories to our bus is required. As there is only one data and address bus, there must be a way of connecting many 1K units to the common bus, while at the same time preventing unaddressed memory from talking to the bus while valid data is being transferred. If this were not possible, a hopeless jumble of bits would be present on the bus each memory cycle, and some hot

**Registered trademark of Teletype Corp.

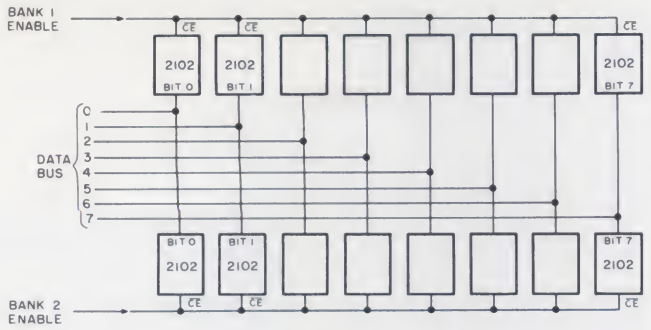


Fig. 2. A 2K memory system. In this simplified memory, the outputs of two "banks" of memory chips are connected to the data bus. Only the memory with the enable line low is allowed to access the bus. The address bus, not shown for clarity, is common to each chip in the memory system. Note that a conflict would occur if both enable lines were in the low state. The address decode circuitry that controls the enable lines is not shown. This memory scheme could be duplicated many times to form a large memory system; however, power consumption would become a factor in memory systems much greater than 8K if 2102 memories were used.

ICs would result. Even the newest TTL designer knows not to *parallel IC* outputs, but how else can multiple kilobyte memory systems function? Referring to Fig. 2, note that the outputs of two chips connect to the individual bus lines in a 2K system. Three-state logic is the *trick* that allows this and other multiple kilobyte memory systems to function.

Conventional TTL

Before discussing the details of Three-state logic, a review of standard TTL, such as the 7400 NAND, and *open-collector* devices like the 7403 is necessary. The output of a single NAND gate is either a ONE or ZERO, depending on the input conditions. (A ONE is defined as an output voltage greater than about 2.4 volts referenced to the 5 volt supply.) There is no other output state possible. Hopefully it can be seen that if two or more regular TTL gates are connected in parallel, damage or an illogical condition will result if one gate's output is low and another high. Obviously the memory system in Fig. 2 could not use regular TTL, or the output lines could not be tied together.

Open-collector logic is a variation on regular TTL. An open-collector gate functions as follows. If both (or all) inputs of a NAND gate are high (ONE), the output is low. This action exactly follows the operation of a regular 7400 NAND gate. However, when any input is low, the output assumes an *open-collector* or *off* state. The open-collector state effectively isolates the output from any following input circuits.

This technique is useful when it is desired to have a gate with many inputs and only one output. A 16 input NAND gate is an example! Note that open-collector logic can only cause a low state, it cannot by itself produce an output high condition. In order to achieve a high output from an open-collector gate a *pull-up* resistor between the output of the gate and the following input stage is required. This resistor is connected to the five volt supply and the open-collector output. Thus, when the gate assumes the open-collector or off state, the resistor pulls the floating output line to a high state. Several open-collector outputs may be pulled-up by a single resistor, as in the case of an eight input NAND gate, illustrated in Fig. 3. This configuration is called a *wired-OR* circuit. (From the logical OR Boolean relationship — any specified input condition causes a corresponding output state.) Referring again to Fig. 3, it can be seen that a high input on any one of the 7405 gates causes the output line to drop to a low state. If all the input gates are low, the output of each is *off* and the pull-up resistor causes the single output line to assume a high state. The use of open-collector logic has two distinct disadvantages: It is slow and noise prone due to the external pull-up resistor, and it is not useful in a bus orientated system. Referring to Fig. 2, the two kilobyte memory, it can be seen that the ideal memory chip must be able to drive an output either high or low without an external resistor and be able to assume an open-collector state. Three-state logic satisfies this requirement.

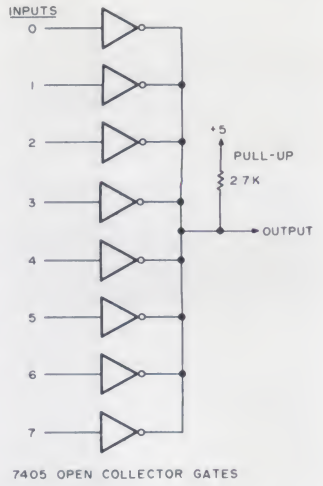


Fig. 3. Eight-input NAND gate, described in the text, is actually formed from eight 7405 open-collector gates with a common output bus and pull-up resistor. When any input goes high, the common output is pulled to a low state. If all inputs are low, the outputs of all gates assume the open-collector "off" state, and the pull-up resistor forces a high output. However, this circuit is noise prone and slow, due to the external resistor.

Three-State Logic

Three-state logic goes a step beyond the earlier described open-collector logic. The output of a three-state package, such as a 2102 memory, can be high, low *or* open-collector. The open-collector off state is controlled by a separate input line, referred to as a *chip enable*, or CE signal. This chip enable line may be activated by either a high or low condition, depending on the chip. The 2102 is enabled by a low signal on the CE line, a high on the line results in the memory output, assuming the open-collector state. Some three-state memory chips, such as the Motorola MC6810, have multiple CE

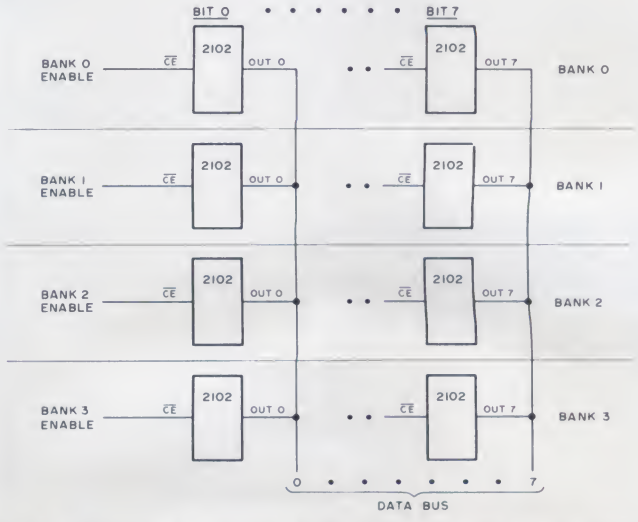


Fig. 4. Simplified 4K memory system, illustrating the use of open-collector logic. All output lines for a given bit are connected together, but only the bank whose chip enable line is low will access the bus. This design is the basis of most microprocessor memory systems using static memory chips.

lines activated by high and low inputs. This particular memory is useful in simple memory systems where the status of microcomputer address lines is used to enable or disable specific memory chips.

Let's examine the role of three-state logic in microcomputer bus orientated systems. The most significant feature of three-state systems is that multiple outputs can be connected to a bus line with only the activated unit controlling the state of the bus. In a typical eight-bit memory system, the chip enable lines for eight 2102s are in parallel. Thus, when the *enable bus* is activated, each chip places its single bit on the bus, resulting in the parallel transmission of a byte. Referring to Fig. 4, a simplified 4K memory, note that each K has its own enable bus, each controlling eight memory chips. Thus, any bank can be placed *on-line* with the bus by activating the enable bus with a low signal. In this example, the other three banks are forced into the open-collector state and do not affect the bus. Obviously some method is required to determine which bank of memory is enabled. This *memory select* logic is beyond the scope of this article, but if the reader is interested in the details of a practical micro memory system, refer to my article, "Short On Memory?" (*73 Magazine*, January, 1977).

Three-state logic is not confined to memory elements. There are several types of three-state inverters, drivers, and gates on the market. I would refer the reader to the major chip manufacturers' data catalogs, as most have a good line of three-state devices.

Hopefully this explanation of three-state logic has resolved some of the misconceptions surrounding this useful logic family, as well as serving as a review of more familiar logic. Its use can simplify many designs and is an essential part of most micro memory systems. Use it in your next design! ■

Letters
to the Editor

from page 73

NO MORE PRAISE?

Just got my first copy of *Kilobaud* and was generally very pleased with it. However, the same thing that bothers me about 73 (1001001) is the content of the letters. I still can't figure out who is it that wants to hear from some guy that "you have the best magazine in the whole world." If he has something else to say, fine, but the heaping of generalized praise is really not a very interesting thing to read about, unless you happen to have written the letter and you like to see your name in print.

I would like to see you divide all your letters into two different sections; one with criticisms and worthwhile instructive comments and one with all of the best magazine letters in it. Then I could forget the second one when I read the magazine.

I know that Wayne loves letters like that, but I hope that his influence on the editorial comment in your periodical is kept to a minimum. It is preferable to inform your readers, not incense them.

The "Glossary" section is a very worthwhile feature. One comment on it while I have the floor. I think that a *baud* is one information unit per second. In systems where the data is strictly binary, a baud *is* one bit per second, but in systems with multilevel coding (such as modems operating at speeds of greater than 1800 bps over voice channels) each timing interval may represent two or three bits. Thus, an 8 phase PSK modem with eight different *tribits* running at 4800 bps is actually using a signalling speed of 1600 baud.

I hope you will include my name in your Sweepstakes. I will be disappointed, indeed, if I don't win the 8800b! I have no system at present, except for my 28KSR and associated modems, but I plan to spend $1 to $2K in the year to come on a computer of some sort. (I subscribe.)

Terry Conboy
Redwood City CA

Since reading your letter, Terry, I find that I've been looking at all the other letters a little differently. I think you'll find out that most of them start off exactly the way yours did. The opening lines will contain a favorable comment . . . and then the writer gets down to brass tacks and says what is really on his mind. Would you like for us to delete those first few lines? No way. We love 'em. – John.

continued on page 115

Hufco PRESENTS THE...

MARK II

FREQUENCY COUNTERS

HOOKUP IS A PIECE O' CAKE with better than 50mv sensitivity, direct connection to the circuit under test is unnecessary in most cases.

FREQUENCIES JUMP OUT AT YOU from the giant ½" readouts.

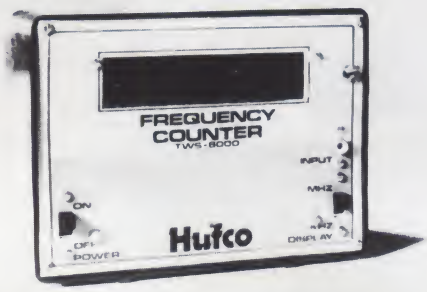
GREATER FREQUENCY RANGE the 60 mHz typical frequency response gives you 80-10 meters plus 6 meters — 50mHz guaranteed.

AND . . .
YOU'LL FIND ENDLESS NEW APPLICATIONS FOR THE "BURNOUT PROOF" MARK II

With the overload protected front end you can use this counter anywhere in a circuit without fear of burnout. Use the Mark II to test: receiver local oscillators, grid dip meters, RF signal generators, audio generators, touch tone pads (when extend installed), micro-processor timing signals, modems, function generators . . . YOU NAME IT!!!

HUFCO QUALITY AS ALWAYS • SAME HI-QUALITY G-10 GLASS EPOXY DOUBLE-SIDED PC BOARDS • SAME TTL IC'S • MORE THAN EVER . . . AMERICA'S BEST BUY IN DIGITAL FREQUENCY COUNTERS!

The TWS MARK II is available in three frequency ranges:
0-50 mHz - 0-250 mHz - 0-500 mHz



RUSH THIS COUPON TODAY!

This is what I've been looking for: A Goof-proof low cost Frequency Counter! Send me:

- ☐ **500 mHz Frequency Counter** - 169.95 kit/199.95 assembled
- ☐ **250 mHz Frequency Counter** - 129.95 kit/159.95 assembled
- ☐ **50 mHz Frequency Counter** - 79.95 kit/99.95 assembled
- ☐ **Information on other handy Hufco Products.**

Enclosed is Check - Money Order - BAC/MC Bankcard OK!

Card No. _____ Exp. Date __/__/____

Name _____

Address _____

City/State/Zip _____

Mail to: Box 375, Dept. K **Provo, Utah 84601**
1603 W. 800 N. **801/375-8566**

H-10

Jim Huffman
Hufco
P.O. Box 357
Provo UT 84601

Automatic Memory Dumper

. . . utility dump program for 6800 users

I think I've experienced some of Jim's frustrations in trying to check memory locations and being held back in speed by the program doing the listing. This is probably an example of overkill in trying to save a few seconds here and there but I'm sure you'll find it interesting. — John.

Program A

READOUT DATA PROGRAM

```
0700   FE   A0   02        Read out S/A in A002 and load A004 with delay value
0703   FF   07   32   B    LDX A002          Get starting MEM address in "X"
0706   BD   E0   BF        STX TEMP          Store contents of X
                           JSR OUT HEX       Jump to MIKBUG output Hex
                                             contents of MEM location in X
0709   FE   07   32        LDX TEMP          Restore X
070C   08                  INX               Increment X
070D   BC   A0   04        CPX A004
0710   2F   1F              BLE C             If not last MEM location do CR/LF
0712   86   0A              LDA B
0714   BD   E1   D1        JSR OUTEEE
0717   86   0D              LDAB              Put out CR/LF
0719   BD   E1   D1        JSR OUTEEE
071C   FF   07   34        TEM STX           Storex TEMP
071F   CE   FF   FF        LDX 64K           Load X with "MAGIC NUMBER"
0722   09              A    DECX
0723   8C   00   00        CPX imm           Is X counted to zero?
0726   27   03              BEQ
0728   7E   07   22        JMP TO A          No, continue decrementing X
072B   FE   07   34        LDX TEM           Yes, get X from TEMP
072E   7E   07   03        JMP TO B          Go-Put out another
0731   3F              C    SWI               Here if last MEM location displayed
0732   XX   XX              TEMP
0734   XX   XX              TEM
```

Here is a nice utility program written in 6800 machine language (assembly language). It occupies only 53 bytes of memory and you can put it anywhere in memory you would like, even though the listing shows it starting at address 0700. The program dumps memory between chosen memory locations and reads out what's in that memory location — does a delay loop, and then goes to the next memory location. The program was developed for debugging long programs such as 3K BASIC, etc.

I found myself trying all sorts of methods to go through a byte-for-byte check of the desired program listing of the program that was in my processor. Obviously, I wouldn't have had this much trouble if I had a hard copy printout with my microprocessor, but there are those of us who must rely on a CRT terminal for our input-output. The delay that is built into the program is long enough to allow you to read the contents of memory from the CRT terminal screen, look down at the paper and confirm whether or not it agrees with the program listing, and about the time it takes you to look back up, you will see a new display.

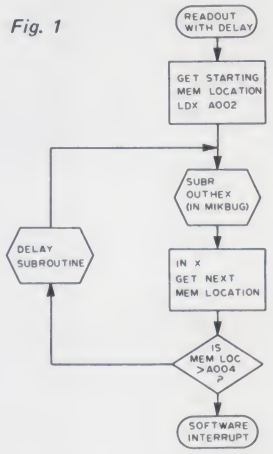
The program was written for SWTPC-6800 microprocessor with the CT1024 terminal. The cursor will automatically advance and the character prints on the next line down the screen. This means if you get a little behind in your checking, all you have to do is look back a few characters, as they will all be printed in a straight line down the screen. This program can be used with a teletype output to give data out with a carriage return and line feed between each data character so that they will appear in a straight program list like an adding machine tape. The starting and ending addresses are loaded into memory position A002 through A005. The program relies on the MIKBUG operating system, ROM. The basic flowchart is shown in Fig. 1, with the program listing and comments shown in Program A.

If you load this program into another position other than beginning address 0700, you will need to change the addresses of the temporary storage located at 0703, 0709, and 071C. You will also have to change the "jump to" addresses located at 0728, 072B, and 072E. Operation of the program is quite simple: 0700 is loaded into A048; the "G" button is pressed; computer responds with carriage return line feed and two hex characters. After a delay, there will be a carriage return line feed and another two characters of data. These are the contents of memory locations starting in A002 and going through whatever is loaded in A004. Address 071F in the program contains the magic number. You load the index register with FFFF for longest possible delay, 0001 for shortest possible delay. You may determine an address that works best for you in your program debugging. Program B shows a typical run of a program. ■

Program B

```
*M A048      FF      07
 *A049       FF      00
 *A04A       FF
 *G
  4F
  2B
  BD
  etc.
```

Fig. 1



Phil Feldman
1722 Brockton Ave. #10
Los Angeles CA 90025

Tom Rugg
1115 N. Beverly Glen Blvd.
Los Angeles CA 90024

Hangmath!

... a new puzzle/game

Phil and Tom have a neat variation of Hangman here . . . and I, for one, really appreciate their attitude concerning non-violence. Let's keep the violence, death and destruction out of computer games, okay? If we (or our kids) really *need that, we can always go turn on a television! — John.*

Hangmath! No, it's not some sort of violent act performed in math classrooms. Actually, it's a new puzzle/game to run on your microcomputer. It's fun to play either by yourself or with a group. This game has good balance requiring equal parts of skill, luck, and deductive ability. It's even educational. For kids learning arithmetic, it will provide a recreational way to sharpen their "number sense."

The program generates a new and challenging puzzle for you each time it is run. Your goal will be to solve the puzzle with a minimum of "errors." When finished, you will be provided with a rating of your performance. Thus, Hangmath can easily be played as a competitive game by comparing performances with your friends.

The Challenge

The program will set up an ordinary multiplication problem of a 3 digit number with a 2 digit number. Initially, the puzzle will look like this:

```
  * * *
    * *
-------
  * * * *
* * * *
-------
* * * * *
```

Your challenge will be to discover which digit from 0-9 each asterisk stands for. The multiplicand and multiplier will be chosen at random by the program except that the leading digit of each will not be zero. The partial products and final product will, of course, be dictated by what was selected. These may have leading zeroes.

Your objective will be to solve the puzzle with a minimum of incorrect guesses. Each guess consists of a trial digit (0-9) and a trial puzzle column (1-5). The columns are considered to be labelled from *right to left* (i.e. the rightmost column is #1 and the leftmost is #5). After each guess, the program checks for "hits." Every occurrence of the guessed digit in the guessed column will be filled into the puzzle. If the trial digit does not appear in the trial column, one incorrect guess will be added to your total. When the puzzle is completely solved (no asterisks remaining) your score will be your total number of incorrect guesses.

Hangmath is Nonviolent

Death by hanging is quite violent and messy. It has always seemed to us to be a cruel and unusual punishment for mere failure in the game of Hangman. In Hangmath we have adopted a much more civilized stance. The program will simply rate your performance in eight categories from "poor" to "excellent" based on your average number of misses over a series of games. This rating will only be meaningful, however, if you do play at least a few games. This allows the good and bad luck of some early hits and misses to "balance out." You should find your performance generally improving as you gain experience (and re-memorize your multiplication tables.)

Of course, it takes a little luck to avoid misses on your first few guesses. But after a few puzzle clues are obtained, it is amazing how a little skill and deduction will pay off in reducing further misses. For example, consider this possible midgame position:

```
    * 2 7
      * 3
-----------
  0 * 8 1
* * * *
-----------
* 4 * * 1
```

Believe it or not, there is enough information here to guarantee solving the rest of the puzzle without a "miss." Note that this is not the same thing as saying that there is only one possible value for each remaining asterisk. Indeed, there is more than one possibility here. However, a cleverly planned series of guesses will insure discrim-

inating among the possibilities without incurring a miss. You might enjoy trying to find it.

How to Run the Program

The program is quite easy to use. The first thing it does is request a random number from the user. This, of course, is used to set the random number seed so that the puzzle changes each time the program is run.

When ready for a guess, the program requests a digit and a column. They are to be input with a comma separating them. If "0,0" is input, this is interpreted as a request to see all of your previous guesses. The program then prints them out for each column. This is handy, for sometimes it is hard to remember exactly which digits you have guessed for which columns.

After each guess, the program prints out the current state of the puzzle and your total number of misses. When the puzzle is solved, your average number of misses per game for the current series of games is computed. Then you are asked whether or not you wish to continue with a new puzzle.

How the Program Works

Fig. 1 was written in MITS 8K BASIC. (Also see Figs. 2 and 3.) String variables are used throughout. The function STR$, which converts a numeric argument to a string, is used in line 345. Also RND, which returns a floating point random number between 0.0 and 1.0, is used in line 150. In that line, a random number obtained from the user is used to reset the seed of the RND function.

The new puzzle is created from line 160 to line 230. A two-dimensional array, A, holds the integer values for each digit in the puzzle. The second argument of A is the row of the puzzle from 1-5. The first argument is the column number from 1-5 increasing from right to left.

```
100 REM THE GAME OF HANGMATH
110 REM WRITTEN BY PHIL FELDMAN AND TOM RUGG — OCT 1976
120 DIM R$(8),F(9,5),A(5,5),P$(5,5),N(5):FOR I=1 TO 8:READ R$(I):NEXT
122 DATA EXCELLENT,VERY GOOD,GOOD,ABOVE AVERAGE,ABOUT AVERAGE
124 DATA BELOW AVERAGE,FAIR,POOR
130 PRINT "HANGMATH — A GAME OF SKILL, LUCK, AND DEDUCTION":PRINT
140 PRINT"PLAY AT LEAST 3 GAMES TO GET AN ACCURATE PERFORMANCE RATING"
150 PRINT:INPUT"    PLEASE INPUT A RANDOM NUMBER";Q:Q=-ABS(Q):Q=RND(Q)
155 G=0:T=0
158 REM GENERATE PUZZLE
160 FOR I=1 TO 5:FOR J=1 TO 5:P$(I,J)=" *":A(I,J)=999:NEXT J,I
170 FOR I=0 TO 9:FOR J=1 TO 5:F(I,J)=0:NEXT J,I:A(1,1)=INT(10*RND(1))
180 A(2,1)=INT(10*RND(1)):A(3,1)=INT(9*RND(1))+1:A(1,2)=INT(10*RND(1))
190 A(2,2)=INT(9*RND(1))+1:N(1)=100*A(3,1)+10*A(2,1)+A(1,1)
200 N(2)=10*A(2,2)+A(1,2):N(3)=A(1,2)*N(1):N(4)=A(2,2)*N(1)*10
210 N(5)=N(1)*N(2):FOR I=5 TO 1 STEP -1:M=10 ↑(I-1):FOR J=3 TO 5
220 Q=N(J)/M:A(I,J)=INT(Q+.001):NEXT J
221 FOR J=3T05:N(J)=N(J)-A(I,J)*M:N(J)=INT(N(J)+.9):NEXTJ
230 NEXT I:A(1,4)=999:A(5,3)=999:B$="  ":N1=0:N5=7:GOTO 360
240 N1=N1+1:GOTO 260
250 PRINT"YOUR INPUT IS NO GOOD,TRY AGAIN"
255 REM GET NEXT GUESS
260 PRINT"DIGIT?,COLUMN?":INPUT D,C:IF D<>0 OR C<>0 THEN 300
270 PRINT:PRINT"        PREVIOUS GUESSES BY COLUMN":FOR I=1 TO 5
280 PRINT"COL.";I" : ";:FOR J=0 TO 9:IF F(J,I)=1 THEN PRINT J;
290 NEXT J:PRINT:NEXT I:PRINT:GOTO 260
300 IF D>9 OR D<0 OR C>5 OR C<=0 THEN 250›0
310 N9=N5
320 IF F(D,C)=0 THEN 340
325 PRINT "YOU GUESSED THAT ALREADY, DUMBO"
330 GOTO 260
340 F(D,C)=1:FOR I=1 TO 5:IF A(C,I) <> D THEN 350
345 P$(C,I)=STR$(D):N5=N5+1
350 NEXT I:IF N9<N5 THEN N1=N1-1
360 REM     OUT1,127:FOR Q=1 TO 100:NEXT
362 REM DISPLAY UPDATED PUZZLE
365 PRINT B$;B$;P$(3,1);P$(2,1);P$(1,1):PRINT
370 PRINT B$;B$;B$;P$(2,2);P$(1,2):PRINT"---------"
380 PRINT B$;P$(4,3);P$(3,3);P$(2,3);P$(1,3):PRINT
390 PRINT P$(5,4);P$(4,4);P$(3,4);P$(2,4);B$:PRINT"----------"
400 PRINT P$(5,5);P$(4,5);P$(3,5);P$(2,5);P$(1,5):PRINT
410 PRINT"NO. MISSES=";N1:IF N5<25 THEN 240
420 PRINT"YOU GOT IT":G=G+1:T=T+N1:V=T/G
430 PRINT"AVERAGE NUMBER OF MISSES AFTER";G;"GAMES IS";V:PRINT
440 PRINT"HOW ABOUT ANOTHER GAME? (1=YES,0=NO)":INPUT Q:IF Q=1 THEN 160
450 Q=V/2:Q=INT(Q):IF Q<1 THEN Q=1
460 IF Q>8 THEN Q=8
470 PRINT:PRINT"YOUR PERFORMANCE RATING WAS ";:PRINT R$(Q):PRINT"BYE"
```

Fig. 1. Program Listing.

```
-------LIST OF VARIABLES FOR HANGMATH -------

A(5,5)    — VALUE OF EACH PUZZLE DIGIT BY ROW AND COLUMN
B$        — BLANK STRING
C         — COLUMN BEING GUESSED CURRENTLY
D         — DIGIT BEING GUESSED CURRENTLY
F(9,5)    — FLAG ON WHICH DIGITS BY COLUMN HAVE BEEN GUESSED
G         — NUMBER OF GAMES PLAYED IN CURRENT SERIES
I,J,Q     — LOOP INDICES, COUNTERS, AND TEMPORARIES
M         — DUMMY DIVISOR IN PUZZLE FORMULATION
N(5)      — ROW BY ROW VALUE OF THE PUZZLE
N1        — NUMBER OF MISSES IN CURRENT GAME
N5        — NUMBER OF SOLVED PUZZLE DIGITS
N9        — NUMBER OF HITS BEFORE PROCESSING CURRENT GUESS
P$(5,5)   — STRING ARRAY OF PRESENT PUZZLE CONFIGURATION
R$(8)     — STRING ARRAY OF PERFORMANCE RATINGS
T         — TOTAL NUMBER OF MISSES IN CURRENT SERIES OF GAMES
V         — AVERAGE NUMBER OF MISSES PER GAME IN CURRENT SERIES


-------HANGMATH ROUTINES-------

100 — SET DATA
150 — INITIALIZE RANDOM NUMBER SEQUENCE
155 — INITIALIZE AND GENERATE PUZZLE
260 — GET NEXT GUESS FROM PLAYER
270 — DISPLAY PREVIOUS GUESSES
340 — PROCESS PLAYER'S GUESS
360 — DISPLAY CURRENT PUZZLE CONFIGURATION
420 — PROCESS GAME TERMINATION
```

Fig. 2. List of Variables and Routines.

A value of 999 is set for those members of A not corresponding to an actual asterisk location. The values of A for rows 1 and 2 are selected at random in lines 170-190. The array N is used to hold the value of each row in the puzzle. Thus N(1) and N(2) are the multiplicand and multiplier. N(3) and N(4) are the partial products and N(5) is the final product. N is computed in lines 190 to 210. Then the values of A for rows 3 through 5 are computed in lines 210-230. (In line 220, .001 was added to the INT argument to avoid some spurious rounding errors.) A few simple tricks are used in this section to set these arrays. We'll let the interested reader dig for them himself.

Adopting it for Your System

When run with MITS 8K BASIC Version 3.2, the code requires about 8½K of memory. We are able to reduce this below 8K by doing the following: removing the trig functions from BASIC, removing the REM statements from the program, reducing the text between quote marks in the PRINT statements. Thus if you have an 8K system, you still can load and

```
RUN
HANGMATH — A GAME OF SKILL, LUCK, AND DEDUCTION

PLAY AT LEAST 3 GAMES TO GET AN ACCURATE PERFORMANCE RATING

   PLEASE INPUT A RANDOM NUMBER
? 3.1416

    * * *
      * *
------------
  * * * *
* * * *
------------
* * * * *

N MISSES=       0
DIGIT?,COLUMN?
? 1,1

    * * *
      * 1
------------
  * * * *
* * * *
------------
* * * * *

N MISSES=       0
DIGIT?,COLUMN?
? 5,1

    * * *
      * 1
------------
  * * * *
* * * *
------------
* * * * *

N MISSES=       1
DIGIT?,COLUMN?
? 9,1

    * * 9
      * 1
------------
  * * * 9
* * * *
------------
* * * * 9

N MISSES=       1
(later in the same game.)
DIGIT?,COLUMN?

    * 5 9
      * 1
------------
  * * 5 9
* * * *
------------
* * * * 9

N MISSES=       1
DIGIT?,COLUMN?
? 2,3

    * 5 9
      6 1
------------
  0 * 5 9
* * 5 4
------------
* * 2 9 9

N MISSES=       7
DIGIT?,COLUMN?
? 0,0

   PREVIOUS GUESSES BY COLUMN
COL.    1:    1    5    9
COL.    2:    3    4    5    6    8    9
COL.    3:    1    2    5    8    9
COL.    4:    0
COL.    5:    0

DIGIT?,COLUMN?
? 7,3

    7 5 9
      6 1
------------
  0 7 5 9
* * 5 4
------------
* * 2 9 9

N MISSES=       7
DIGIT?,COLUMN?
? 5,4

    7 5 9
      6 1
------------
  0 7 5 9
* 5 5 4
------------
* * 2 9 9

N MISSES=       7
DIGIT?,COLUMN?
? 6,4

    7 5 9
      6 1
------------
  0 7 5 9
* 5 5 4
------------
* 6 2 9 9

N MISSES=       7
DIGIT?,COLUMN?
? 4,5

    7 5 9
      6 1
------------
  0 7 5 9
4 5 5 4
------------
4 6 2 9 9

N MISSES=       7
YOU GOT IT
AVERAGE NUMBER OF MISSES AFTER       1    GAME IS       7

HOW ABOUT ANOTHER GAME? (1=YES, 0=NO)
? 0

YOUR PERFORMANCE RATING WAS GOOD
BYE
```

Fig. 3. Sample Run.

execute the program.

The program is better run on a CRT (or "TV typewriter") than on a terminal with hard copy because the frequent reprinting of the puzzle tends to consume much paper. If you have a CRT which does not scroll, line 360 should be used to control clearing the screen and "homing up." This avoids having the puzzle broken between the top and bottom halves of the screen. Removing the REM on line 360 will accomplish this for the system we used. The OUT 1,127 sends an ASCII 127 (DEL or rubout) to output port #1. This port was our CRT and the rubout was the control character to clear the screen and home up. The loop on line 360 is simply a delay timer to insure that the paged memory of the TV terminal is cleared before any new output is sent to it.

Getting the "Hang" of It

Hangmath affords opportunities for many different game plans. We have watched good players using all kinds of initial strategies. Some like to guess digits from the rightmost columns to the left. Others go left to right and still others try all over the lot. Some even guess a particular digit in all columns first. As you play, you will tend to develop your own style and tricks. Soon, you'll get the hang of it. With a little luck, you may even solve a puzzle with no errors.

If you find yourself reaching for your calculator, consider yourself a full-fledged member of the calculator generation. Try solving the puzzles without it. You'll have fun remembering the good old days of long-hand arithmetic and multiplication tables.

So the next time you have a few friends over gawking at your micro, treat them to a few rounds of Hangmath. For once, you may enjoy "hanging" your system. ■

Letters
to the Editor

from page 108

WHO'S AFRAID OF THE BIG BAD MATH?

I really enjoyed issue #1 of *Kilobaud*. I read better than three out of four articles instead of reading one out of an entire magazine as I do with some other computer hobbyist magazines. I especially liked Art Child's article on software exchange suggestions.

I do want to take exception to your answer to Webb Simmons letter on page 67. You state your opinion that most of your readers understand some math but don't like to use it. While I might agree with this statement for many other magazines (like 73) I don't feel that it is true about computer freaks.

Since most of the computer hobbyists have come from the ranks of the professional hardware or software people I think it is fair to assume that they do have a good background in mathematics. I'll bet that over a third of *Kilobaud's* readers have had at least one course in differential calculus! I don't believe, however, that these people will be afraid of math. An important point is that much of the dislike of math stems from the drudgery associated with simple adding, subtracting, multiplying, or dividing numbers! This busywork need not bother the computer hobbyist since the computer will do all his dirty work for him.

Mathematics is both a tool to use in order to pursue an end and it is an end in itself. The hobbyist needs tools. He needs formulas for interest calculations, formulas to determine which sort is more efficient, formulas to generate random numbers, etc. Isn't it better, though, to tell him how the formulas were derived and why they work rather than to give him a program and say merely, "Here it is. It works."?

The computer hobbyist is usually motivated by a strong sense of curiosity and by a desire for fun. (One reason why we see so many games for computers!) Math can be fun in and of itself. Many people enjoy crossword puzzles or cryptograms and math puzzles are no less enjoyable. Many of them can be worked with nothing more than simple arithmetic while others require elementary algebra. One word of warning though – Math puzzles are habit forming, like peanuts – you can't work just one!

How does this apply to *Kilobaud*? I would like to see articles or perhaps a monthly column devoted to math puzzles. These articles could propose puzzles appropriate for computer solution and give enough background for their solution. There could even be contests for the best computer solution.

I expect that someone will object that this *theoretical* material isn't practical and doesn't belong in *Kilobaud*. I'm reminded of a supposedly true story I once read about a man who decided to devote his efforts to solving problems in pure mathematics. If a problem had the slightest chance of being applied to something practical then he wouldn't even look at the problem. He had to give up his attempt. Every time he would prove a new theorem or solve a new problem some joker would come along and find a practical application for it! This situation is not uncommon in mathematics. I'll give only one example. James Clerk Maxwell described mathematically the nature of electromagnetic waves thirteen years before Heinrich Hertz was able to produce them. I wonder if Maxwell thought anyone would ever find a use for his theories.

For over fifty years amateur radio operators have been making important contributions to the electronics and radio arts. Perhaps solving impractical problems and developing techniques for their solution will be one of the ways amateur computerists will contribute to their own art.

I hope other readers feel as I do and will write in to express their views. I'm not afraid of mathematics!

Glen Charnock
Oxnard CA

I'm certainly open to input on this subject. – John.

Grant Runyan
1146 Nirvana Road
Santa Barbara CA 93101

Now – BASIC for the 8008 – Even!

It always distresses me to hear about people who started building 8008-based systems several years ago (particularly the Mark-8) and never got them finished, or retired them, because they couldn't run any decent software. Well, as Grant points out here, there is definitely some decent software for it now! – John.

There must be hundreds, maybe even thousands, of Mark-8s and other hobby computers using the 8008 microprocessor lying around and not getting much use. The principal reason is that their owners have had to write all their own software in machine language which is a wonderful exercise but gets very laborious before much gets accomplished. There is a higher level language available for the 8008 which really puts that ubiquitous little chip into the category of powerful microprocessors.

The language is called SCELBAL which stands for SCientific ELementary BAsic Language and also just happens to sound like the first part of Scelbi Computer Consulting, Inc., which is its purveyor. It has been patterned after a commonly used higher level language referred to as BASIC. SCELBAL was written specifically for the 8008, but it can be run on any other computer whose instruction set has the instructions of the 8008 as a sub set. This would include the 8080 and Z-80.

The chances are that no one would want to use SCELBAL on anything other than an 8008 permanently because there are faster and better BASICs, but for learning how to use BASIC, SCELBAL is tops because it has the most important commands and functions of BASIC, and the commented source listings are so complete and straightforward that one can really get to understand what makes BASIC work.

In the Beginning . . .

After the first introduction of the 8008 to the general public, the proliferation of microprocessors has been nothing short of furious. Many who started building a system based on the 8008 were probably not even well along until the 8080 became available. In some cases the original project was set aside in favor of a new start with the 8080. Chances are that some who started an 8080 project never really got a system running until they jumped to the Z-80 or something else.

Apparently there is never going to be an end to the opportunity to switch systems, so it might be better to go back and get the old original project running even if it's only going to be a toy for junior to operate. The learning to be gained is tremendous, and the usefulness of an 8008 system with a good program should not be underestimated.

An Interpreter

SCELBAL, like any BASIC, is an interpretive language, not a compiler. That means that the language program essentially processes each line or statement in the source code of the higher level syntax and then executes the directive before going on to the next line or statement. It does this by calling on machine language routines that perform the various functions as soon as it has been determined which job is to be accomplished. A compiler, however, merely compiles, that is, it produces machine code, and what it produces is executed at a later stage.

An interpreter such as SCELBAL has everything required to create and execute a program residing in memory at one time. Thus, once the SCELBAL program itself has been loaded into memory, the operator can prepare and execute many different kinds of programs in a short time span. This is particularly true for inexperienced programmers as they can almost instantaneously be notified of syntax errors and immediately make corrections to the program being created on an on-line, interactive basis.

Features

SCELBAL, like other good BASICs, is designed to operate in a *calculator* mode or operate in a stored-program mode. For instance, if one typed in the statement

```
PRINT 2*2 + 3*3
```

followed by a carriage-return, the value 13 would immediately be returned on the output device. The calculator mode may be used to operate as a powerful calculator in which values for variables may be stored. If one entered the statements

```
LET A = 2
LET B = 3
```

and

```
PRINT A * A + B * B
```

the device would again return

13 upon the receipt of a carriage-return.

In the stored-program mode the user simply inserts a line number in front of each statement. A whole series of statements may then be arranged to form a program. When it is desired to execute the steps in the program, a special executive RUN command is issued, and the program is executed one step at a time.

SCELBAL contains a floating point package which handles addition, subtraction, multiplication, division and raising to integral powers for positive and negative numbers. When inputting information or specifying values, the user may use fixed point notation for numbers in the range plus or minus 0.999999 to 999999. Numbers smaller or greater than this must be stated in floating point format, such as +0.234567E-10 or -654321E+12. The maximum powers that SCELBAL can handle is ten to the minus or plus 38th. All calculations in the floating point package are maintained to twenty-three significant binary bits in the mantissa with binary rounding when the calculations exceed that number of bits. This means that results are accurate to six significant decimal digits. A seventh digit is sometimes returned but cannot be counted on for accuracy. Parentheses may be used to any depth to group or nest mathematical statements.

The Executive

The executive portion of SCELBAL has five major commands available to the operator which are defined and explained briefly below.

SCR indicates the SCRATCH command. It effectively clears out any previous program stored in the program buffer along with any previously user-defined variables. It is used in preparation for entering a new high level program into the program storage area.

LIST is the command that does just that! It causes the contents of the program buffer to be displayed or *listed* on the system's output device so that it may be reviewed by the operator.

RUN directs the interpreter to begin operations and execute the program stored in the program buffer.

SAVE may be used to direct the system to save a copy of the program stored in the program buffer on the system's external bulk storage device. A program saved using this command can later be restored for further use by using the command presented next.

LOAD directs the program to read in a copy of a program from an external bulk storage device.

SCELBAL Statements

SCELBAL recognizes the following types of statements, and they result in selected operations being performed.

REM indicates a comment or REMark which is to be ignored as far as the interpreter is concerned. Information on a line prefaced by REM is intended only for the use of programmers and may be used to document a program.

LET is used to set variables equal to a numerical value, another variable, or an expression. The implied LET is also available. Thus, the simple statement, X = 3, has the same effect as LET X = 3.

IF ... THEN allows the program to make decisions. If the conditions of the IF part are met, the program proceeds to do the instructions following the THEN. When the conditions of the IF part are not met, the THEN instructions are ignored, and the program proceeds to the next step. The symbols >, <, and = as well as sensible combinations of them are used in IF ... THEN statements.

GOTO directs the program to jump to a specified line number in a program.

GOSUB is a statement that directs the program to perform another statement or group of statements as a subroutine. It is followed by the line number of the first statement of the subroutine. When the GOSUB command is used, the program saves the line number of the next step in the program so that the RETURN statement will bring the saved line up next.

RETURN indicates the end of a subroutine. When it is used, the program will return to the next line number following the GOSUB directive which was used to call the subroutine. SCELBAL permits multiple nesting of subroutines (up to eight levels) within a program.

INPUT is used to direct the interpreter to wait for an operator to input information. After the information has been received, operation of the program automatically continues. While the computer is waiting for an input, typing a CTRL-C will cause an escape back to the executive portion.

PRINT is used to output information from a program. This statement allows mixed types of output on the same line (numerical values and alphanumeric messages), and it allows the option of providing a carriage-return and line-feed after outputting information or suppressing those functions. Also see remarks on TAB for formatting.

END is a statement used to designate the conclusion of a higher level program in the program buffer.

DIM is an optional statement available to users who have sufficient memory to adequately support it. It is used to specify the formation of a one-dimensional array in a program. Up to four arrays having a total of up to 64 entries are permitted.

The Functions

The power of SCELBAL is further enhanced by the addition of seven functions which may be used within statements. These functions are discussed below.

INT returns the INTeger value of the expression, variable, or number requested as the argument.

SGN returns the SiGN of the variable, number, or expression. If the value is greater than zero, the value +1.0 is returned. If the value is less than zero, the value -1.0 is returned. The value 0 is returned when the expression or variable is zero.

ABS returns the ABSolute value (magnitude) without regard to the sign.

SQR obtains the SQuare Root of the expression, variable, or number.

RND produces a semi-psuedo RaNDom number in the range of 0 to 0.999999.

CHR is a special CHaRacter function which will cause the ASCII character corresponding to the decimal value of the argument to be displayed. It must be used only in a PRINT statement. The reverse function is specified by placing a dollar sign ($) immediately after a variable in an INPUT statement. An input of a character would then cause the decimal value of the ASCII code to be input.

TAB is a function for use in a PRINT statement which causes the display device to space over to the column specified.

There is a higher level language available for the 8008 which really puts that ubiquitous little chip into the category of microprocessors.

UDF is the User Defined Function for routines which the user may wish to implement in machine language. If one is so inclined, he can add any functions that he might like and call them with UDF followed by an identifying number as an argument. The programs are limited only by the programmer's ability and available memory space.

System Requirements

SCELBAL is designed to run in a system having a minimum of 8K bytes of read and write memory. In an 8K system, the program, leaving out the optional DIMension capability, provides about 1,250 bytes of memory for storage of the user's higher level language program. It is recommended that users opting for the DIM capability have 12K of memory. Three pages or 3/4K of memory are required for the part of the program containing the DIM routines. The program is assembled so that those instructions reside on pages 55, 56, and 57. Instructions are given for locating the DIM object code, the program buffer, the I/O routines in any available memory space, but the main body of the program resides on pages 1 through 32 octal.

Page zero of memory is left for the user's monitor or bootstrap routines and is also recommended for location of user input and output programs. However, these routines may be put wherever the operator wishes and has memory space. Clear instructions are provided in the text for making all necessary adjustments.

The user of SCELBAL must write his own input and output routines because no author could possibly write routines that would suit everyone's equipment. For parallel I/O devices the routines are trivial and merely consist of machine language that instructs the values found in the A register to be output as ASCII characters, and ASCII values from input devices to be input to register A. For serial devices the programs are somewhat more complicated, but the programmer has registers B, H, and L available for output routines and registers B, D, and E for input. Similar user-devised programs must be written to make the LOAD and SAVE commands operate. The addresses of the routines are simply inserted in specific memory locations to cause the program to call whichever I/O routine is desired.

The execution speed of SCELBAL, while slow compared to higher level programs that are designed to run on large computers, is surprisingly good. The 8080 version is about ten times as fast as the 8008 due to the relative speeds of the CPUs. The execution speed of the 8008 can be almost doubled if one uses the 8008-1 and increases the clock speed. However, even on an 8008 based unit, the execution time is quite tolerable. After all, who needs instantaneous response from a hobby computer? We have all become accustomed to a slight waiting period from calculators, and the slight delay makes us appreciate the thousands of steps that sometimes must be taken to execute a single line of BASIC.

Conclusion

SCELBAL is published in book form by Scelbi Computer Consulting, Inc., 1322 Rear—Boston Post Road, Milford, Connecticut 06460. The retail price is $49.00. You are not advised to attempt to implement SCELBAL from someone's borrowed tape. The annotated source code and explanation of operation of each routine along with flowcharts is the great value of the publication. Using SCELBAL is a learning situation, and you will learn better with a text. It is well worth the price.

The publication lists the object code for the 8008 in octal right along with the source listing. Another identical source listing is printed with the object code in octal for the 8080. In other words, the program is identical for both the 8008 and the 8080, but the two CPUs require different object codes to accomplish the same results. The authors explain that the program is far from the most efficient for the 8080 because it does not use any instructions which are not also instructions for the 8008. It is not as efficient as it might be for either CPU because of not using any of the RST (restart) instructions that are located on page zero for both chips. As explained earlier, page zero is reserved for the user's routines.

With a monitor system that allows one to enter programs from a keyboard in octal, it will require from eight to twelve hours to load the program depending on one's adeptness with the keyboard. The chances of doing all this without a mistake are rather slim, and so, probably several more hours will be required to check the entries. The program is quite unforgiving and often will hang up completely because of an error of just one bit of a single byte. Naturally, no one wants to do all this loading to run SCELBAL just once. You will have to use a magnetic tape system or a paper tape punch and reader to record the program and load it again. The Scelbi company has not attempted to prepare a magnetic tape for sale because of the lack of standards. They do have for sale a paper tape of the object code available for registered purchasers of the book for $25.00. The Hexadecimal Paper Tape Format promulgated by Intel Corporation for use in their INTELLEC MCS * (*TM) is the compromise format decided upon among the many possibilities. This seems to be the most familiar to industry and university users where the majority of the requests for such tapes are reported to be coming from.

Scelbi company has instituted a publication named *SCELBAL UPDATE* which is being sent to all registered purchasers of the program. At this writing there have been three issues. The purpose of this publication is to keep users informed of the appearance of any *bugs* that might appear (there have been a few) and how they might be corrected. Interesting user-created programs are solicited and published with a small honorarium for the authors. The third issue announces that string capabilities for SCELBAL will soon be made available.

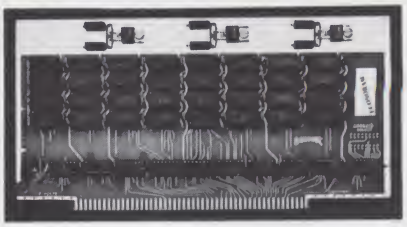
In summary, SCELBAL is the fastest and best high level program for the 8008. It is not the best or the fastest in existence, nor is it the most complete for the 8080, but it is the most carefully explained and the easiest to alter and modify for the user's purposes, and it offers the most learning experiences of them all. Since most hobbyists are not making real time landings on the moon, who needs so much speed? So, let's see some of those Mark-8s and other 8008s come down off the shelf and get to doing their thing. ■

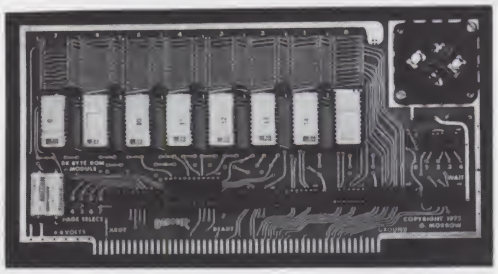
The execution speed of SCELBAL, while slow compared to higher level programs that are designed to run on large computers, is surprisingly good.

GODBOUT

BILL GODBOUT ELECTRONICS
BOX 2355, OAKLAND AIRPORT, CA 94614

TERMS: Add 5% to order to cover shipping & handling. Cal res add tax. You may place Mastercharge® and BankAmericard® orders thru our 24 hr order desk (415) 562-0636. Sorry, no CODs. **We carry computer related items from parts to peripherals.** For details, request our free flyer.





ECONOROM™ $99.95

OVER TWO YEARS OF EXPERIENCE SELLING 4K X 8 BOARDS HAS SHOWN US **EXACTLY WHAT YOU WANT IN A MEMORY BOARD KIT.**

--**YOU WANT LOW POWER** TO STRETCH YOUR POWER SUPPLY. ANY COMPANY CAN CLAIM LOW POWER; NOT EVERY COMPANY OFFERS A SPEC TO BACK IT UP. WE **GUARANTEE** CURRENT CONSUMPTION **UNDER 750 MA**, WITH THE AVERAGE BOARD FALLING BETWEEN 600 AND 650 MA.
--**YOU WANT S-100 BUSS COMPATIBILITY**...AND ECONORAM IS FULLY COMPATIBLE.
--**YOU WANT CLEAN AND UNAMBIGUOUS DATA TRANSFER**...WHICH IS WHY WE BUFFERED OUR ADDRESSES, DATA LINES, AND OUTPUTS LONG BEFORE THE OTHER GUYS CAUGHT ON.
--**YOU WANT A FAST BOARD.** BY USING MEMORIES GUARANTEED AT 450 NS WORST CASE OVER THE FULL TEMP RANGE, WE CAN **GUARANTEE** THIS BOARD TO RUN AT **ZERO WAIT STATES (500 NS OR BETTER).** A 450 NS MEMORY ALSO ALLOWS FOR ANY PROPAGATION DELAYS IN SUPPORT CIRCUITRY.
--**YOU WANT QUALITY, AND WE DELIVER IT**: FROM THE EPOXY GLASS, DOUBLE SIDED, PLATE THROUGH BOARD...TO THE LOW PROFILE SOCKETS... THE OPTIMIZED THERMAL DESIGN...THE DIPSWITCH ADDRESS SELECTOR... THE LOW POWER SCHOTTKY SUPPORT ICS...THE GUARANTEE WE OFFER ON ALL PARTS USED IN ECONORAM...AND MORE.
--**YOU WANT LOW COST.** BECAUSE OF OUR PURCHASING POLICIES AND QUANTITY BUYING, WE CAN STILL DELIVER A BOARD OF THIS QUALITY FOR UNDER $100. YOU CAN PAY LESS, AND YOU CAN GET LESS. BUT **IF YOU WANT THE BEST COMBINATION OF VALUE AND ECONOMY...ECONORAM GIVES YOU BOTH.**

ALSO AVAILABLE ASSEMBLED, TESTED, AND WARRANTEED FOR ONE YEAR FOR $129.95. DISCOUNTS ON ALL KITS AVAILABLE FOR GROUP PURCHASES. G-4

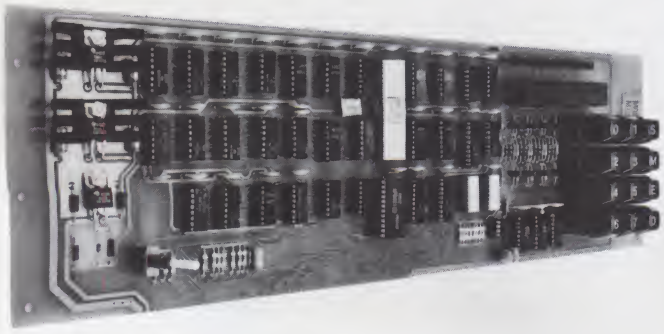
ECONOROM™

GIVE YOUR CPU A FRIEND: A HOME FOR PROGRAMS AND ROUTINES!

OUR BASIC ECONOROM BOARD IS 4K X 8 WORTH OF ERASEABLE ROM, WHICH YOU CAN **PROGRAM WITH ANY PROGRAM OR ROUTINE YOU WANT** (OR HAVE US PROGRAM IT FOR YOU). SIMILAR FEATURES AND SAME QUALITY AS OUR ECONORAM---AND ALSO **VERY LOW POWER:** 5V @ ½A, -12V @ 150 MA. WE OFFER SEVERAL OPTIONS:

BASIC ECONOROM BOARD (4K X 8).....**$179.95**
SMALLER ECONOROM BOARD (2K X 8).....**$135.00**
BIGGER ECONOROM BOARD (8K X 8; HOLDS 8K BASIC).....**$269.95**
8080 SOFTWARE BOARD (THIS IS OUR 4K BOARD, PROGRAMMED WITH EDITOR, ASSEMBLER, AND MONITOR ROUTINES FOR THE 8080...A VALUABLE FIRST STEP IF YOU'RE TRYING TO GET AWAY FROM MACHINE LANGUAGE PROGRAMMING. INFO PACKAGE AVAILABLE THAT DESCRIBES THE FUNCTION OF ALL SOFTWARE FOR $2.95 (REFUNDABLE WITH ORDER).....**$189.95**

Goodbye, microcomputer, it was fun...
Hello, Minicomputer!



This is the heart of an S-100 buss compatible minicomputer system. Upgrade your present machine with this board or form a custom system around it using S-100 buss peripherals.

There are a lot of good microcomputers on the market. But that's all they are: microcomputers. Difficult to use, frustrating for program development. Full of blinking lights and CPUs that go to sleep when you halt the program.

The **Sigma-100 Minicomputer CPU Board** (with integral front panel) allows you to gain control over your machine. Run your program, or if you wish, step it at any rate from 1 to 1000 steps per minute ("Slow Step"). By simply stopping the machine, you may examine and alter processor registers, memory locations, and I/O devices--there's special firmware to keep the CPU from going to sleep ("Control Halt"). You can monitor all of the above during execution of the program as well. Everything is front panel controlled by your fingertips through a 12 pad keyboard; octal data reads out on unambiguous 7 segment displays.

At last, a machine where you can examine, alter, and monitor every funtion of the CPU/front panel and its operation **in real time**. Edit or modify your program while you run it. . .think of what this means in terms of extra productivity, reduced frustration, and greater speed.

The implementation matches the concepts, with onboard regulation, sockets that prevent soldering damage to ICs, low power Schottky ICs that save energy, buffers for full isolation. . .in other words, quality.

We invite you to send for our flyer, which gives additional details. If you're curious how we did it, send for the board's documentation package ($5.00).

MORROW'S Micro-STUFF BOX 6194 ALBANY, CA 94706

Available by mail and at many computer stores.
Kit form: $250 Assembled/tested: $325 M-7

Dr. Lance A. Leventhal
138 South Acacia Ave.
Solana Beach CA 92075

Microprogramming

. . . an insight into microprocessor design

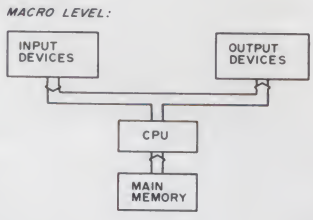
With regard to subject material we would have to classify the following in the heavy *area . . . but the presentation balances that fact out. There is almost always confusion when someone mentions* microprogrammed *computers around us* microcomputer *types. Lance does a good job of explaining the difference between the two. — John.*

In 1951, it was first proposed that a computer could execute its instructions by breaking them down into a series of smaller instructions. That is, a computer inside the main computer would execute programs which would transform the instructions of the main computer into the required actions. This concept is called *microprogramming*. We will investigate what microprogramming means (and why it is so confusing) and why the user of personal computers should be concerned with it at all.

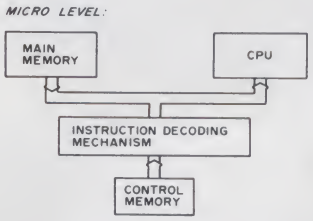
Why Worry About Microprogramming?

In fact, the computer user seldom worries about instruction decoding or microprogramming. The computer manufacturer defines what the instruction set means and can do; the user must live with what the manufacturer provides.

Then why do we want to talk about microprogramming? One reason is just to understand how the computer works. But microprogramming can offer more than understanding since it is a key factor in making computers more powerful and easier to use. The more functions that the computer carries out at the microprogram level, the more powerful and flexible its instruction set will be. A major difference between the Intel 8080 and the Zilog Z-80 is in the microprogramming. Improved addressing, hardware multiplication and division, block input/output transfers, and multi-word instructions are typical of the features that can be added through microprogramming. For now, the computer user can only compare processors and hope that the manufacturers will provide improvements. But eventually, microprogramming will become simpler and either the users or manufacturers will be capable of providing features far beyond those in today's small computers. Microprogramming will be the key to one of the next big steps in personal computers.



The real-world computer transforms input data into output results according to a program stored in main memory.



The controlling computer transforms instructions from the main memory into internal computer activities according to a program stored in the control memory.

Fig. 1. Microprogramming — two levels of computing.

The Meaning of Microprogramming

But what does microprogramming really mean? The idea is to consider the normal control function of the computer as the execution of a series of instructions, i.e., to imagine the CPU as itself containing a computer which has as its ultimate task the execution of the instructions of the overall computer. A computer which executes

instructions in this way is said to be *microprogrammed*; the instructions which the control section executes in order to fetch and decode the instructions of the overall computer are called *microinstructions*. The concept of microprogramming is confusing since it involves using one computer as part of another computer; even experienced computer specialists find it difficult to separate programs from microprograms, instructions from microinstructions, and main memory from microprogram memory. To the beginner, microprogramming often seems like working with mirrors.

But note that, if a computer can control the functions of a nuclear reactor, a factory machine, or a cash register, there is no reason why it cannot control the functions of another computer. In fact, the instruction cycle of a computer can logically be divided into a series of sequential actions — just the sort of problem a computer can easily solve. A typical series for an addition instruction would be:

1) Place the contents of the program counter on the memory address bus.

2) Add 1 to the contents of the program counter (updat-

(a) INTEL 8080

OPERATION	INSTRUCTION CODE REGISTER	IMMEDIATE
ADD	10000RRR	11000110
ADD WITH CARRY	10001RRR	11001110
SUBTRACT	10010RRR	11010110
SUBTRACT WITH BORROW	10011RRR	11011110
AND	10100RRR	11100110
EXCLUSIVE OR	10101RRR	11101110
OR	10110RRR	11110110
COMPARE	10111RRR	11111110

NOTE: RRR is a 3-bit register code.

Bits 3, 4 and 5 determine the operation as follows:

BIT 5	BIT 4	BIT 3	OPERATION
0	0	0	ADD
0	0	1	ADD WITH CARRY
0	1	0	SUBTRACT
0	1	1	SUBTRACT WITH BORROW
1	0	0	AND
1	0	1	EXCLUSIVE OR
1	1	0	OR
1	1	1	COMPARE

(b) MOTOROLA 6800

OPERATION	INSTRUCTION CODE
SUBTRACT	80 – F0
COMPARE	81 – F1
SUBTRACT WITH BORROW	82 – F2
AND	84 – F4
BIT TEST	85 – F5
EXCLUSIVE OR	88 – F8
ADD WITH CARRY	89 – F9
OR	8A – FA
ADD	8B – FB

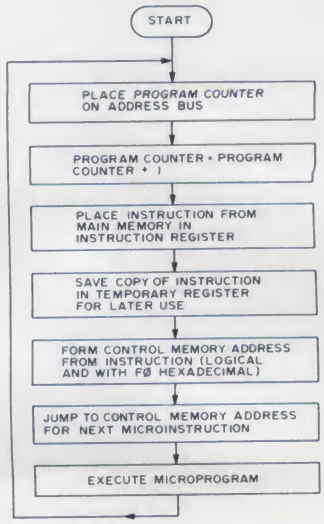
Here the first four bits determine the addressing mode and accumulator to be used. The least significant four bits determine the arithmetic or logical operation.

Table 1. Meaning of bits in Arithmetic and Logical instructions.

ORGANIZATION OF CONTROL MEMORY

ADDRESS (HEX)	CONTENTS
00	FETCH SEQUENCE WHICH GETS INSTRUCTION FROM
●	MAIN MEMORY, SAVES COPY FOR LATER USE, AND
●	FORMS A JUMP ADDRESS FROM MOST SIGNIFICANT
0F	HEX DIGIT
10	DECODING SEQUENCE FOR INSTRUCTION 1
●	
1F	JUMP 0
20	DECODING SEQUENCE FOR
●	INSTRUCTION 2
●	
2F	JUMP 0
●	ETC.

Fig. 2. Using an instruction to find the decoding sequence in Control Memory.



Flowchart of instruction fetch procedure.

ing it for the next instruction fetch).

3) Place the data from memory in the instruction register.

4) Determine the sources of the data for the arithmetic unit (ALU).

5) Send the ADD function code to the ALU.

6) Determine the destination for the result from the ALU.

Much of this series would be the same for other instructions. For example, only step 5 would be different for a subtraction, comparison, or logical instruction. The first three steps would be common to all instructions and many other steps could be shared by a wide variety of instructions. For example, indexing or other addressing methods would be the same regardless of the operation involved. The idea of microprogramming is to make each step in the series into a microinstruction. The CPU always starts each cycle by executing the fetch microinstructions, uses the instruction obtained from memory to choose which microinstructions will then be executed, and concludes by returning to the fetch sequence.

A microprogrammed computer therefore has two levels. The CPU fetches instructions from the main memory in order to perform its overall system tasks. It also fetches microinstructions from a *control memory* in order to decode and execute the instructions from the main memory (the overall instructions are sometimes called *macroinstructions*). Each instruction will generally involve several microinstructions; however, the total number of microinstructions in the control memory is usually quite small since the total number of distinct instructions is not very large and many microinstructions can be shared. Fig. 1 contains block diagrams of the two levels, each of which has all the usual functions of a computer.

An Implementation of Microprogramming

How does the CPU choose the right microprogram? Fig. 2 shows a common method. Here the 8-bit instruction consists of two 4-bit hexadecimal digits; the most significant digit is the operation code. The instruction decoding mechanism saves a copy of the instruction for later use and forms a jump address in control memory from the operation code. For example, if the operation code is 6, the next microinstruction to be executed will be in control memory location 60 hex. Each decoding sequence ends with a jump back to the fetch sequence at address 0. Note that operation code 0 is a *no-operation* since it immediately returns to the fetch sequence. Remember that Fig. 2 shows the microprogram in the control memory, not the user program in main memory.

The microprograms for particular instructions could then examine the rest of the instruction. For example, all arithmetic and logical instructions might differ only in the last four bits. The common microprogram would send these four bits to the ALU and thereby determine the function to be performed. Table 1 shows the meaning of instruction bits in the arithmetic and logical instructions on the Motorola 6800 and Intel 8080.

In the case of conditional jump instructions, the last four bits could determine which status flags were to be checked and whether the value should be zero or one. The microprogram would then simply provide a return to the fetch sequence if the conditions were not met. Table 2 shows the actual meaning of instruction bits in the conditional jump instructions on the Motorola 6800 and Intel 8080.

Advantages and Disadvantages of Microprogramming

We should note that microprogramming does not

(a) INTEL 8080

CONDITIONAL JUMP	INSTRUCTION CODE (BINARY)
JC-JUMP ON CARRY	11011010
JNC-JUMP ON NO CARRY	11010010
JZ-JUMP ON ZERO	11001010
JNZ-JUMP ON NO ZERO	11000010
JM-JUMP ON MINUS	11111010
JP-JUMP ON POSITIVE	11110010
JPE-JUMP ON PARITY EVEN	11101010
JPO-JUMP ON PARITY ODD	11100010

Bits 0, 1, 2, 6, and 7 identify this group of 8 instructions. The other 3 bits determine which flag is the condition and whether that flag should be a logical 1 or a logical 0. Bit 3 is the logic level and bits 4 and 5 determine the flag as follows:

BIT 5	BIT 4	FLAG
0	0	ZERO
0	1	CARRY
1	0	PARITY
1	1	SIGN

(b) MOTOROLA 6800

CONDITIONAL JUMP	INSTRUCTION CODE
BCS-BRANCH IF CARRY SET	00100101
BCC-BRANCH IF CARRY CLEAR	00100100
BEQ-BRANCH IF = ZERO	00100111
BNE-BRANCH IF NOT EQUAL ZERO	00100110
BLT-BRANCH IF $<$ZERO	00101101
BGE-BRANCH IF $\geq$ZERO	00101100
BLE-BRANCH IF $\leq$ZERO	00101111
BGT-BRANCH IF $>$ZERO	00101110
BLS-BRANCH IF LOWER OR SAME	00100011
BHI-BRANCH IF HIGHER	00100010
BVS-BRANCH IF OVERFLOW SET	00101001
BVC-BRANCH IF OVERFLOW CLEAR	00101000
BMI-BRANCH IF MINUS	00101011
BPL-BRANCH IF PLUS	00101010

Bits 4, 5, 6, and 7 (the most significant hexadecimal digit) identify the branch instructions (00100000 is the unconditional branch). Bit 0 is the desired logic level for the conditional flags. Bits 1, 2, and 3 determine the condition as follows:

BIT 3	BIT 2	BIT 1	CONDITION
0	0	0	NONE
0	0	1	C + Z
0	1	0	C
0	1	1	Z
1	0	0	V
1	0	1	N
1	1	0	N $\oplus$ V
1	1	1	Z + (N $\oplus$ V)

Here C is CARRY, Z ZERO, N NEGATIVE, and V OVERFLOW.

Table 2. Meaning of bits in Conditional Jump instructions.

necessarily have anything to do with microprocessors despite the confusing similarity in names. Large computers like models of the IBM 360 and 370 series and minicomputers like the Burroughs 1700 and Hewlett-Packard 2100 are microprogrammed. In fact, most modern CPUs, regardless of their size, are microprogrammed.

What do we gain from microprogramming? The answer is that we get the usual advantages of using a computer rather than simple circuitry. A microprogrammed CPU is more flexible, can be customized more easily to particular applications, and can be changed or corrected more readily than can a hard-wired CPU. These advantages are particularly important for microprocessors where chip design is a difficult and time-consuming process; a microprogrammed processor can be corrected, customized, or redesigned by changing the microprogram in the control memory rather than by changing the entire device, a process which could easily take one or two years.

The disadvantages of microprogramming are its difficulty and slowness. Microprograms are hard to write because of the amount of detail involved and the lack of software and hardware support. Furthermore, a microprogrammed computer is basically slower than a hard-wired computer because of the overhead involved in fetching and decoding microinstructions. Ultra-high-speed computers therefore must be hard-wired. An obvious problem with microprogramming is that each instruction cycle will generally involve the fetching and execution of several microinstructions. Clearly the microinstruction fetch will have to be quite fast to avoid slowing down the entire computer. In fact, microprogramming was seldom used until recently when the cheap, high-speed memories needed to implement it efficiently became available.

Usually the computer manufacturer writes the microprograms and places them permanently in read-only memories. In microprocessors, this control read-only memory is part of the LSI chip. The user does not have access to the microprograms which are part of such CPUs as the Intel 8080 or Motorola 6800. A few computers allow the user to change the microprograms; such computers are said to be *microprogrammable*. Popular microprogrammable computers include the Burroughs 1700, Varian V-73, Hewlett-Packard 2100, Interdata 8/32, and Nanodata QM-1. A few microprocessors are microprogrammable — examples are the National IMP-16, Western Digital MCP 1600, and Intel 3000. Microprogramming is presently difficult, tedious, and requires special equipment; significant advances will be necessary to make this power conveniently available to the user.

Instruction decoding and execution is the center of computer activity. The basic purpose of the computer is fetching, decoding, and executing instructions. Microprogramming is a flexible method for decoding instructions through a second level computer. Although few users of personal computers will do any microprogramming, an awareness of how it works is often helpful in understanding the internal operations of the CPU. Microprogramming also offers great future potential for making personal computers more powerful and easier to use.■

References

Dollhoff, T. L., "Microprogrammed Control for Small Computers," *Computer Design*, May 1973, pp. 91-97.

Jaeger, R., "Microprogramming: A General Design Tool," *Computer Design*, August 1974, pp. 150-157.

Wyland, D. C., "How To Design Your Own Microprocessor," *Electronic Design*, September 27, 1975, pp. 72-78.

Glossary

Tim Barry

ARRAY: (1) A group of keys or indicators which are operated upon as a unit (i.e., an input switch array or an output status array). (2) A data structure in which each element is identified by one or more unique position indicators. In mathematics, arrays are often operated upon as units by applying special arithmetic rules. In many programming applications the term simply refers to an area assigned to store program data.

EXPONENTIAL: An exponential function is defined by the general equation: $F(x) = x^t$, meaning that the value x is raised to the T power. Thus $10^5 = 10*10*10*10*10 = 100{,}000$. Equations which contain exponential terms tend to increase or decrease very quickly.

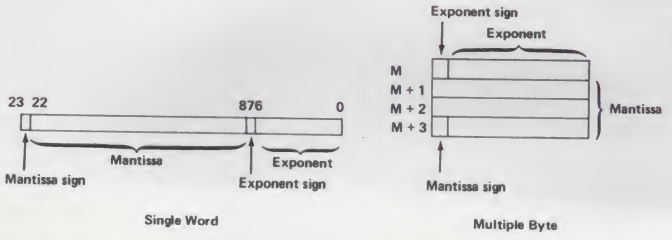
FLOATING POINT: Floating point notation is a convention used to represent a wide range of real numbers in the computer. Each number is considered to consist of a *mantissa*, M, and an *exponent*, E. (The exponent is sometimes called the *characteristic*.) Any number N in radix (base) R can be represented as the product of the mantissa multiplied by the radix taken the exponent power:

$$N = M * R^E$$

The number 123456 in base 10 can be represented in floating point notation as any of the following:

$$123456 * 10^0$$
$$12.3456 * 10^4$$
$$.123456 * 10^6$$

Most computer floating point systems treat the mantissa as a signed binary fraction and the exponent as a signed integer power of two. All input values are *normalized* to match this format. Having all numbers in the same format makes it easier to set up floating point arithmetic operations. One of the most common floating point systems is to use a 24-bit mantissa and an eight-bit exponent. In an eight-bit microcomputer this means that each floating point number will require four bytes of storage. Using this scheme, numbers in the range $-.9999999 * 10^{-38}$ to $.9999999 * 10^{38}$ can be represented, and this is adequate for most applications.



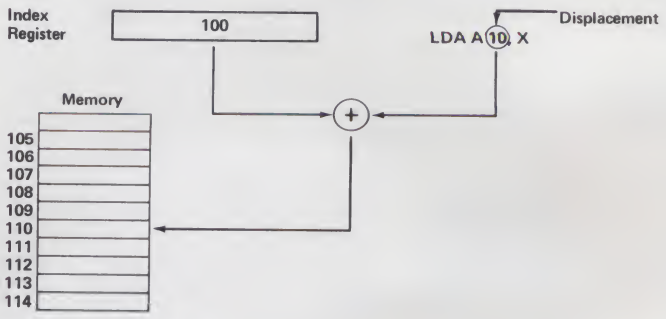
FOREGROUND/BACKGROUND: A system's software organization which allows both direct and deferred execution of user programs. Foreground programs are executed interactively under user control. The user maintains complete control over editing, execution, inputs, outputs and other program functions. Programs in the background are executed on the basis of time available to the computer system. All required execution data is set up prior to submitting the job for execution. Once a job is submitted to the background, user control is usually limited to checking program status (running, waiting, printing, etc.) or aborting the job. Once the job is complete the user can examine the output data.

Foreground/Background processing originally developed as an extension of time-sharing. It allows fast, user interactive time-share services to coexist with large, slow batch jobs. Most modern large computer systems support some form of foreground/background operating system.

INDEXED ADDRESSING: A computer instruction which uses indexed addressing refers to the contents of a memory location whose address is computed by adding a displacement included with the instruction to the contents of an index register. For example, in the Motorola 6800, the instruction sequence:

```
LDX #100
LDA A 10,X
```

will load the A accumulator with the contents of the location specified by adding the displacement of 10 to the 100 in the index register. Indexed addressing is a very convenient way to handle manipulations of data in tables. The index register is initialized to the start of the area containing the data. The data can then be sequentially accessed by modifying the index register contents.

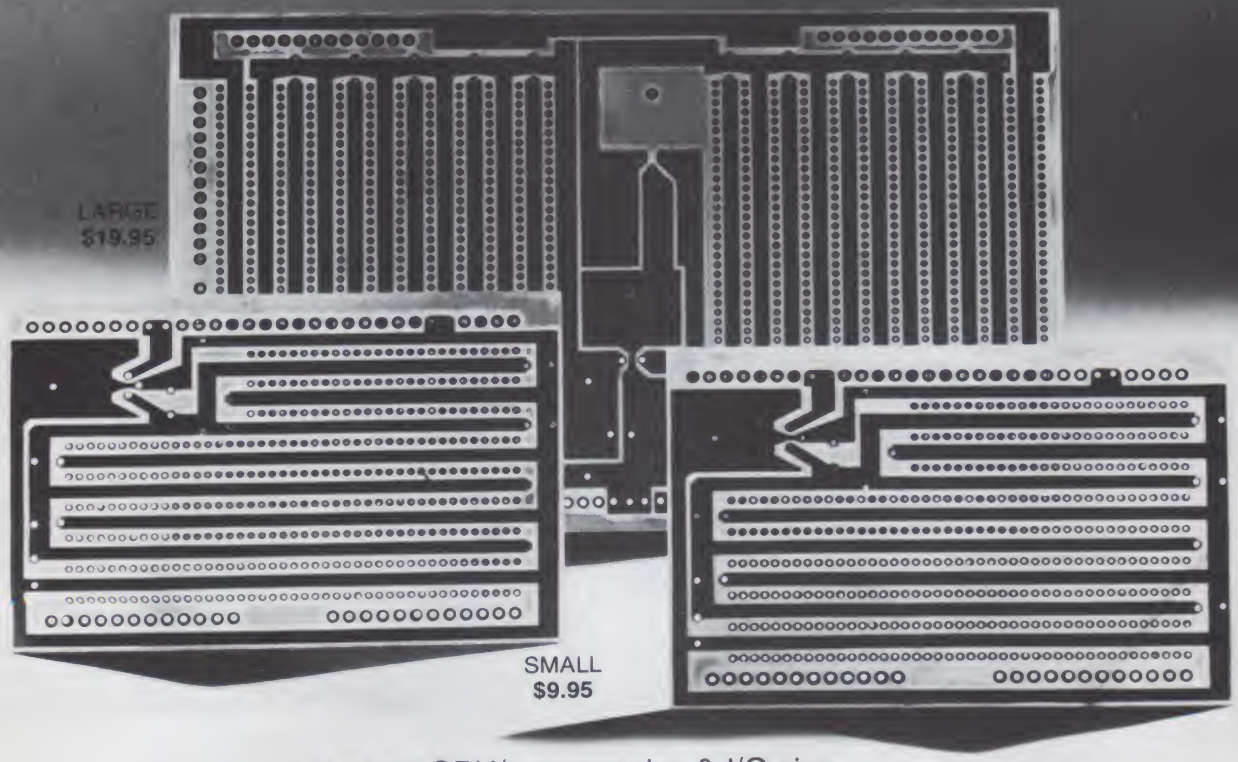


INDEX REGISTER: An index register is a CPU register whose contents can be used to form an indexed address. In most computers the index registers can also be used for temporary data storage and other program operations.

6800?

Now!! Have One Your Way

The 6800 system owner can now have his best ideas in hardware on a buss compatable card designed to mate the SWTPC 6800 system.



- 2 sizes: CPU/memory size & I/O size
- Will accept 14, 16, 24 and 40 pin connectors
- Test and/or interface connections on top
- 2 on-board regulator locations (1 on small board)
- Short, low inductance power and ground
- Use with wire wrap
- Use with wiring pencil

SEND MONEY ORDER, CHECK OR BANK AMERICARD # (We prefer Bank Americard)

Personal Computing Company

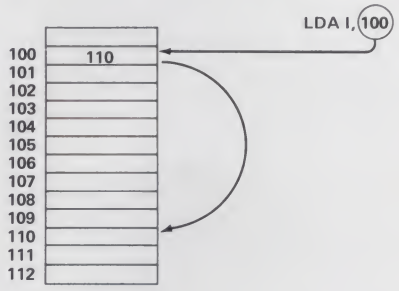
3321 Towerwood Drive, Suite 107
Dallas, Texas 75234

DEALERS INVITED

INDIRECT ADDRESSING: A computer instruction which uses indirect addressing refers to a memory location whose contents are to be used as the address of the memory location to be used in the operation. For example, the load indirect instruction

LDA I,100

would use the contents of location 100 as the address of the location whose contents are to be loaded into the accumulator.



Most commonly used microprocessors do not have a true indirect address capability. Instead, they use *register indirect addressing.* This addressing mode uses the contents of a CPU register to specify the address of the memory location to be used in the operation.

INTEGER: An integer is a positive or negative counting number; it has no fractional part. Computers represent all data internally as binary integers, and all data manipulations require the hardware or software to keep track of whether the individual values are to be interpreted as numbers, characters, instructions, or anything else.

REAL NUMBER: The mathematical definition of a real number includes all integers, repeating fractions, and non-repeating fractions such that $-\infty < R < +\infty$ (where R is the radix, or base). In computer usage a real number is considered to be any constant or variable represented in floating point notation.

REAL TIME CLOCK: Real time programming requires that a computer system execute its operations within a time period constrained by events occurring in the outside world. A real time clock is a piece of hardware which interrupts the processor at fixed time intervals. These intervals are usually set to between 1 and 1000 ms, and they are very carefully controlled. By counting these interrupts the computer can keep track of elapsed time and use this timing information to determine when to perform its control operations.

STRING: A string is a group of data elements stored in sequential memory locations and treated as a unit during certain program operations. *Character strings* are the most commonly encountered type of string, and they are usually composed of ASCII printing and carriage control codes.

TERMINAL: A computer/user interface composed of an input device through which the user inputs commands and data to the computer and a display device upon which the computer displays information to the user.

TRACE: (1) Printed circuit board interconnections formed by selectively etching the copper surface during board fabrication. (2) A software diagnostic technique used to follow program execution step by step to determine where an error is occurring. A running trace usually displays the contents of all CPU registers as each instruction is executed, thereby enabling the user to determine where values are not changing as predicted.

from page 13

program. In general, it will not be as great as machine level data saving programs. Fig. 2 illustrates the potential limitation to data rate when using BASIC to save data.

The same programming technique can be applied to saving numerical data. The only difference is that the numerical value must first be converted to a string, then scanned and transmitted to the tape device. The program in Example 7 illustrates this technique.

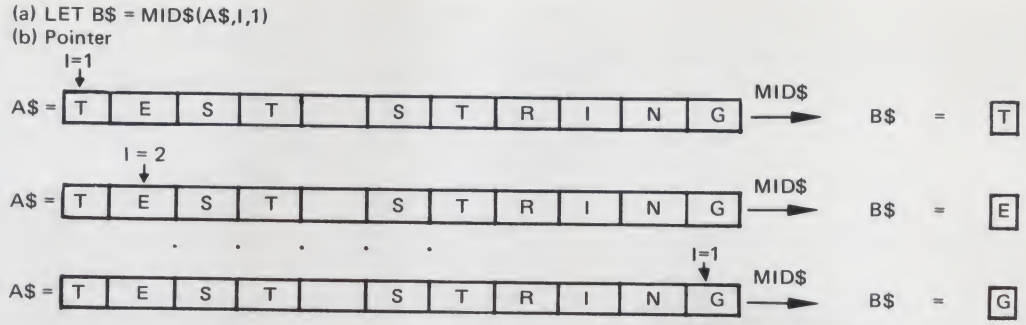


Fig. 1. (a) The BASIC statement used to separate the individual characters. (b) A diagram showing how incrementing the pointer variable "I" is used to scan the character string.

The program in Example 8 demonstrates how to read a saved numerical value back.

The programs presented can be converted to subroutines so that they can be called conveniently during the execution of a program.

There are other techniques for storing data – some more efficient and able to achieve higher data rates. For example see the article "Data Recording with the Altair" in the November, 1976, issue of the *MITS Newsletter.*

If you have some ideas on the subject, data recording, please drop us a note. We would be happy to share them with other BASIC users. The address for corresponding is:

BASIC FORUM
305 Clemson Drive
Tyler TX 75703.

```
  5 REM NUMERIC VALUE SAVE PROGRAM
 10 LET B = 3.14159
 15 REM CONVERT VALUE TO STRING
 20 LET A$ = STR$(B)
100 DIM T(50)
140
 .
 .        SAME AS STRING SAVE PROGRAM
 .
999
```

Example 7.

```
  5 REM NUMERIC VALUE LOAD PROGRAM
 10
 ...
 ...      LINES FROM STRING LOAD PROGRAM
110
120 LET B = VAL (A$)
130 PRINT B
140 END
```

Example 8.

```
1 Print"Publisher's Remarks"
10 No End
Run
```

from page 3

sales. Then there are two more girls who prepare and address the direct mail letters ... one to sort and bundle them, and so forth.

These letters have to be written ... I do most of that ... then made ready to print by the art department ... and printed by the tens of thousands by the printing department. They also print most of the books we publish, the *Kilobaud Newsletter*, QSL cards, and things like that. It's enough work to keep two shifts going seven days a week in the press room.

Add to those already mentioned a second crew for *73 Magazine*, more to prepare our books and tapes, bookkeeping, a shipping department to pack and mail out current issues, back issues, books, etc., and you can see where we have a staff of over 60 people ... and need more. A small group can produce a magazine if they have most of the work done by subcontract, but it takes a lot of people to do everything involved and launch a fair-sized subscription promotion to boot.

That's a brief description of the system we use to bring you *Kilobaud* ... if you find yourself in the vicinity, please stop in and say hello. We have visitors almost every day and enjoy meeting our readers and getting their candid opinions.

MAKE PAPERS PAY THEIR WAY

Preparing a paper for presentation at a symposium or other affair is a lot of work. Essentially it is the same as writing an article for publication ... and it should earn its way just as an article would.

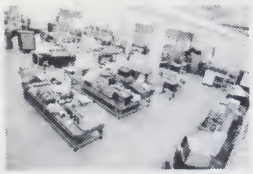
Presumably, if a subject is of enough interest to draw a group of people to hear your talk, it will also be of enough interest to warrant publishing. Obviously only a tiny fraction of the people who would be interested will be able to attend the talk, so the rest will miss out if you don't have the paper published.

Next comes the question of where to have your paper published. Though payment for it as an article may not be of first importance, publishing it where you get no payment is the same as giving that publication a present of $150 to $300 or so. Few of us are wealthy enough to be that philanthropic. And what possible reason is there for such a philanthropy? The publishers of "nonprofit" magazines are just as much commercial enterprises as anyone else.

The more logical approach is to pick a publication which will reach the readership that will benefit most from your material ... and which will also pay top dollar for the material. This narrows things down a lot ... essentially your choice is between *Kilobaud* and *Byte*. I believe *Kilobaud* pays a lot more for articles than *Byte*, but the difference between the two magazines is more than that ... it has to do with the level of your paper. If you've written a paper for the expert ... for the PhD, then try *Byte* first. If you're aiming at the beginner and less sophisticated computerist, try *Kilobaud*.

Keep in mind that publication of hardware ideas may well preserve your rights to them if you want to seek a patent. The same may be true of software, if the legal end of that is ever satisfactorily sorted out by the courts.

Letting your hard work be published for free is just making a gift of it to someone else ... is this what you want? Your paper is worth money ... so why give it away? ■


COMPUTER WAREHOUSE STORE

DEPT: K • 584 COMMONWEALTH AVENUE • BOSTON, MA • 02215 • 617-261-2701 • VISIT US: 9-9 WEEKDAYS; 9-6 SATURDAYS

ONE DAY SHIPMENT

PERIPHERALS FOR MICROSYSTEMS

ALL an ASR 33 is and MORE!

RS232 INTERFACE, QUIET OPERATION, 10 CPS PRINTER, BUILT-IN PAPER TAPE READER/PUNCH, ELECTRIC TYPEWRITER KEYBOARD WITH ADDITIONAL 10 KEY NUMERIC PAD, YOUR CHOICE OF FRICTION OR SPROCKET FEED, LIGHTED PLATEN AREA FOR EASY READING, STANDARD PAPER AND TAPE, SUPPORTED BY OLIVETTI.

$950 + SHIPPING 165 lb

TALLY T132

7 x 8 DOT MATRIX IMPACT PRINTER HAS A SINGLE LINE DYNAMIC MEMORY AND A UNIVERSAL INTERFACE TO ACCEPT PARALLEL DATA, FORMS TO 14-7/8 IN. WIDE, SIMPLE PRINTING MECHANISM USES 132 SOLENOID HAMMERS AND TWO STEPPER MOTORS FOR 100 LPM, 132 COLUMNS, 64 CHARACTERS

$450 + SHIPPING 150 lb

DIGITRONICS D507.....$95 + 400 lb SHIPPING

PAPER TAPE READER, HEAVY DUTY PS, 3 MUFFIN FANS, POWER CONTROL PANEL, CIRCUIT BOARDS, RELAYS IN CABINET

KDI ADTROL AR-21...$95 + $25 SHIPPING

ELECTRO OPTICAL PAPER TAPE READER WITH 110V PS, STEPPER MOTOR, 250 CHAR/SEC, FAN IN TABLE TOP HOUSING

IBM 731 I/O WRITER $750 + $25 SHIPPING

8½"PLATEN, PINFD, EBCDIC, U/L CASE, DUAL CLR RIBBON, 115B

KITS ★ HIGHLIGHTS FROM OUR WIDE SELECTION ★

IMSAI 8080 MICROKITS

8080A KIT 5 SLOT	$599
8080A KIT 22 SLOT	651
4K MEMORY KIT	139
PIC-8 PRIORITY INTERRPT	125
SERIAL I/O KIT	125
PROM 4-512 KIT	165
UCRI-1 KIT	59
CABLE A KIT	18

STANDARD INPUT/OUT INTERFACES, MANUALS, SOCKET SETS

SWTPC 6800

- 512 BYTES of ROM
- RS232 or 20 mA
- SERIAL INTERFACE
- 4K of RAM **$395**

4K MEM	$100		
MPA	145		
MPC	40	MPB	$ 40
MPE	14.95	MPD	35
MPM	65	MPF	30
MPP	35	MPMx	35
MPS	35	MPL	35
MPMb	14.50	MPAb	14.50
MPCb, MPSb, MPLb, EACH			9.50
CONNECTOR SETS MPU/MEM			2.50
CONNECTOR SETS INTERF.			2
4KBA	5	GT61	99
AC30 AUDIO INTERFACE			79.50
PP40 PRINTER			250
CT 1024 TERMINAL KIT			275
CTP	15.50	CTS	39.95

ALL SWTPC UNITS ARE KITS

VIKING 100 PIN CONNECTOR HEAVY DUTY	$ 3

LEAR SIEGLER ADM-3A W/ Cursor Control

- 12" CRT
- 24 LN X 80 CHAR
- RS232
- 20 mA LOOP

$875

ICOM MICROFLOPPIES

PLUG COMPATIBLE FOR: $100 BUS....FD2411..... $1095
SINGLE DRIVE FD2402.....$ 649

ICOM FLOPPIES:

FF36-1 FRUGAL	$1195
FF36-2 DUAL FRUGAL	1395
360-58 BLT; INTFC 8080	300
S171 POWER SUPPLY	250
FD360-2-5 DUAL SYSTEM	3000

KIM-1 6502	$ 245
KIM-2 4K	179
KIM-3 3K	289
MANUALS PACKAGE	15
TARBELL AUDIO CASSETTE KIT	120

INTERSIL INTERCEPT JR.	$281
12K RAM	145
ROM/PROM BOARD	74.65
YOU ADD MEMORY CHIPS	
SERIAL I/O	81.50
AUDIO VISUAL BOARD	125
SCAMP KIT	$99

FROM NATIONAL SEMICONDUCTOR

KEYBOARD KIT	95
SMOKE SIGNAL BROADCASTING, 16K RAM	595

SPECIAL DISCOUNTS! ON KITS & ASSEMBLED UNITS

SAVE UP TO 20% OFF KIT PRICE WHEN A PERIPHERAL IS PURCHASED AT THE SAME TIME.

PERIPHERAL PRICE OVER $900 ➡ 20% OFF KIT
PERIPHERAL PRICE OVER $250 ➡ 10% OFF KIT
PERIPHERAL PRICE OVER $95 ➡ 5% OFF KIT

GREEN PHOSPHOR VIDEO MONITOR $150 +25 SHIPPING

STANDARD 1V P TO P COMPOSITE VIDEO INPUT, 16 MHz BAND WIDTH, RASTER SCAN 12x12x13" WITH POWER SUPPLY, VIDEO AMPLIFIER, DRIVING CIRCUITRY VENTILATION MUFFIN FANS, 7x9" HORIZONTAL VIEWING, CAPABLE OF 24 LINES x 80 CHAR., ANTIGLARE ¼" ETCHED GRADIENT DENSITY FACE PLATE, P39 GREEN PHOSPHOR, ON/OFF BRIGHTNESS CONTROLS, 115Vac, 60 W ... TRULY A COMMERCIAL UNIT BUILT TO WORK IN A DEMANDING ENVIRONMENT.

DATAPOINT 3300-200 THERMAL PRINTER

PARALLEL PRINTER WITH ADDITIONAL CIRCUIT BOARDS TO PROVIDE SERIAL INTERFACE, PRINTS UP TO 30 CPS, 100Vac PS USES WIDELY AVAILABLE NCR PAPER, 96 CH, ASCII, 80 COL, CRT COMPATIBLE 5x7 DOT MATRIX, SOLID STATE WITH LESS THAN 25 MOVING PARTS.

$475 + $25 SHIPPING

DATAPOINT CONSOLE PRINTER

BOTH UNIVAC AND SINGER BUILT THESE PRINTER MECHANISMS WHICH OPERATE AT 30 CPS FROM A ROTATING WHEEL. 65 CHAR. USES STANDARD FORM FEED PAPER. PINWHEEL IS INTERCHANGEABLE.

$395 + SHIPPING 285 lb.

UNIVAC 0769-06 PRINTER MECHANISM ONLY ... $195
INCLUDES MOTOR/PRINT WHEEL +SHIPPING 75lb.

KLEINSCHMIDT 311..... $250 + 75 lb SHIPPING

THIS 30 CH/SEC DRUM PRINTER SITS IN A SOUND-PROOF ENCLOSURE. 64 CHAR, PARALLEL INPUT, 80 CHAR/LN

TECHTRAN 4100..... $595 + 35 lb SHIPPING

TAPE CASSETTE DRIVE. CAN RUN DIRECTLY FROM TERMINAL INDEPENDENT OF CPU. FULL EDIT CAPABILITY.

TO ORDER EQUIPMENT

1. ENCLOSE CHECK FOR FULL PRICE PLUS SHIPPING CHARGES (KITS - ADD $5 IF UNDER $100; $10 IF OVER)
   BANKAMERICARD & MASTERCHARGE ACCEPTED
   SEND CARD NUMBER, EXPIRATION DATE, INTERBANK #
2. CLEARLY IDENTIFY SHIPPING ADDRESS
3. DESCRIBE ITEM BY MODEL NUMBER

ALL MERCHANDISE WARRANTEED

WRITE FOR OUR CATALOGUE

INCLUDES DESCRIPTIONS OF OUR COMPLETE LINE OF KITS & UNITS, REVIEWS OF OVER 150 BOOKS, LIST OF NEW & SURPLUS PARTS AND ARTICLES - "ALL ABOUT PERSONAL COMPUTERS"

$1 C-27

Jennifer at the electronic palette.

Computerized Babysitter

... graphic display for kids (and parents)

Alfred S. Baker
2327 So. Westminister
Wheaton IL 60187

I'm seriously considering throwing in a bonus for any articles which include a photo of a cute young lady such as Al's article does. The best part, though, is the fact that Al is using his home system to get his kids turned onto computers . . . or at least make sure they don't grow up with a fear or misunderstanding of them. — John.

Processor Technology Corp. makes the VDM-1, a terrific TV text display interface board for the ALTAIR bus. This board will generate 16 lines of 64 characters each on a standard black and white TV (with simple modifications described in the documentation for the board). Also, any character displayed can be white on a black background or black on a white background and all 128 of the possible ASCII characters print something: upper case, lower case, and a huge selection of special characters.

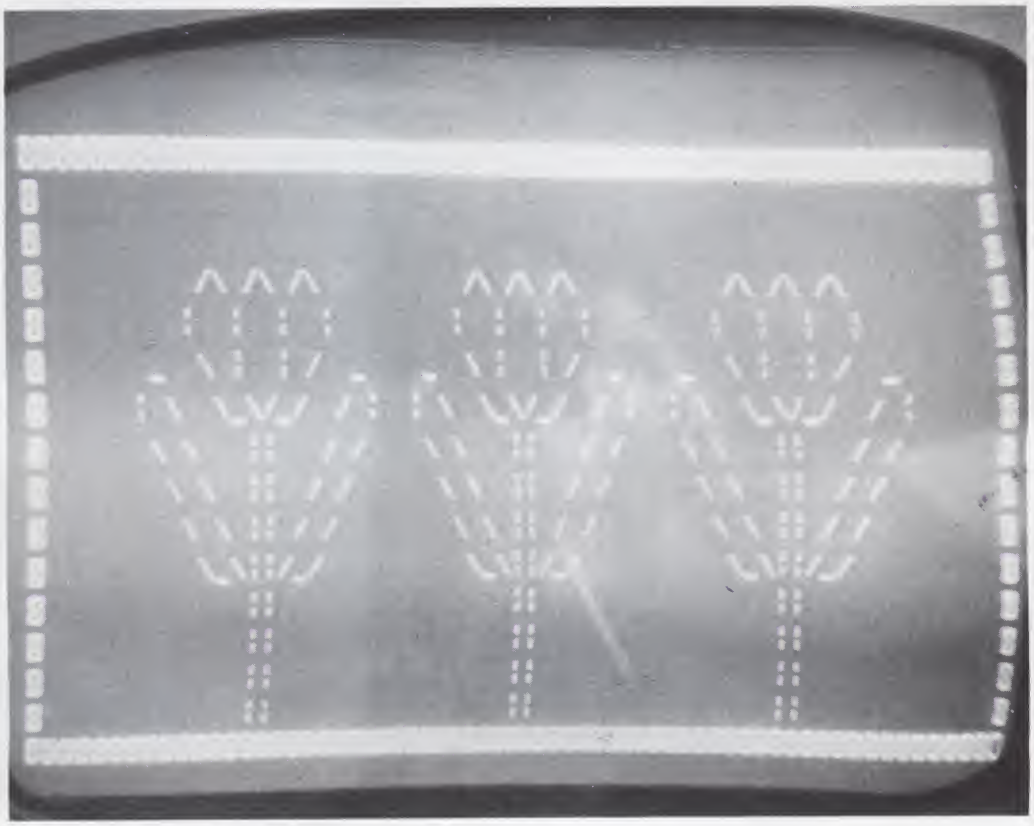
So far, so good. Unfortunately, my four year old daughter, Jennifer, wanted to draw pictures and I had spent my budget. Therefore, rather than purchase a TV graphics display board, I wrote the program DRAW. The program was written using Processor Technology's version of the 8080 assembly language. I had a ball writing it. Jennifer has a ball using it.

The program CALLs three external routines which I added to the basic operating system, or monitor, originally written by Processor Technology and provided by IMS Assoc., Inc. as part of their Imsai 8080 system. They should be replaced by equivalent routines in your system. Having a repertoire of such routines available for use is one of the things which converts a home computer from toy into tool. Once you invent your own personal wheels, you should not have to reinvent them each time they are needed.

KEYIN controls and accepts input from my keyboard. The input character is placed in the accumulator.

FILEO outputs selected portions of memory to a standard audio cassette using Morrow Micro-Stuff's cassette interface (Kansas City Standard). The routine must be provided with the address of the area to be written in the HL register pair and the length of the area in the DE

Notice the arrows on the eight keys surrounding the S key and the optional white key to the right.



A simple picture using the "splat" and other special characters for detail.

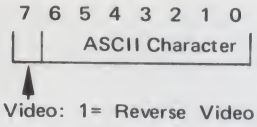


Fig. 1. VDM-1 Character.

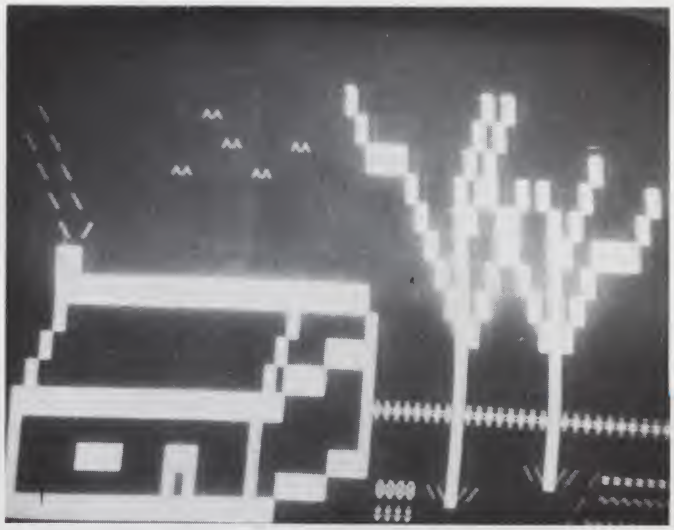
register pair. The routine takes this information, a checksum it generates, and packages both into a file header followed by the data. FILEI inputs from cassette into memory. The location and size of the data read in is taken from the file header on the cassette. If the data read in does not verify against the checksum stored in the header, this routine enters an error signal loop.

The keyboard controls of DRAW are simple to use. From the photograph of the keyboard it is possible to see the arrows drawn on the eight keys surrounding the S key. I use these eight keys in all games or other applications requiring a toggle or stick style control. To move in a particular direction, hit the appropriate arrow. In DRAW, these toggle keys move the current location marker around on the screen. This marker is represented by the VDM-1 box character □ (or decimal 0 internally). As this marker is moved around on the screen, the current drawing character (described below) is left behind.

As previously mentioned, the VDM-1 lets the user place any character on the screen in either white on black (standard video) or black on white (reverse video). This is controlled by the *high* or 7 bit in each byte in the VDM's memory (see Fig. 1). The 1024 bytes (characters) of memory used by the VDM-1 to display the 16 lines of 64 characters is addressed as part of the computer's memory. In my machine this area is from location CC00 hexadecimal to location CFFF hexadecimal. Every byte in this area which has the 7 bit on is displayed as black on white. The remainder is displayed as white on black.

When DRAW is first entered, the drawing character is the reverse video space. (Jennifer calls it the splat.) It is used to draw the house and the trees in the third photograph. At any time this character can be changed by hitting the ESC or escape key followed by the new character. To convert this character to reverse video, the V key is then struck.

One modification I made to my keyboard was to have it output 8 bits rather than the standard 7 bit ASCII code. The keyboard was provided with several user definable keys. One of these keys now outputs the eighth bit. When pressed along with another key, it makes that character show up on the screen in reverse video. I painted the top of this key white to signify its function. Therefore the V key is available for those who cannot make a similar modification to their own keyboard.



One of the author's attempts at "art".

Besides the toggle keys, the ESC key, and the V key, there are four other keys used by DRAW. Two of these keys are located near the toggle keys and are used for reading and writing pictures to cassette. Although their use is not fatal to the current picture, it can be inconvenient. Therefore, both require the simultaneous pressing of two keys to avoid accidental use. The control-S key (CTRL or control key and the S key) will store the current picture on cassette and the control-R key will read a picture from cassette to the screen.

The final two keys are fatal to the current picture. Therefore not only do they require two keys to be pressed together, but the keys are located at some distance on the keyboard from any other keys used by DRAW. Control-L is used to leave DRAW and return control to the monitor. Control-K clears the screen.

I hope this small program has given you some insights into the artistic possibilities of the VDM-1. Now your family can stretch its imagination in your own home electronics art center. ■

DRAW Assembly Listing

```
ADDRESS  MACHINE CODE  SEQ. NUM.  LABEL  OP-CODE  OPERANDS  COMMENTS
1780     CDD417        0020       DRAW   CALL     CLR       USE CLEAR THE SCREEN
                                    *                        ROUTINE.
1783     CDC6CA        0040       MAIN   CALL     KEYIN     GET KEYBOARD INPUT.
1786     4F            0060              MOV      C,A       SAVE INPUT IN REG. C.
1787     218317        0080              LXI      H,MAIN    PLACE ADDRESS OF MAIN
178A     E5            0100              PUSH     H         ON STACK. THUS WHEN ALL
                                    *                        SUB-ROUTINES ISSUE THE
                                    *                        RET INSTRUCTION, EXECUTION
                                    *                        WILL CONTINUE AT MAIN.
178B     FE0B          0120              CPI      'K'-40H   IF INPUT WAS CONTROL-K,
178D     CAD417        0140              JZ       CLR       'CALL' CLEAR THE SCREEN.
                                    *                        RETURN IS TO MAIN.
1790     FE1B          0160              CPI      1BH       IF INPUT WAS ESCAPE,
1792     CAB517        0180              JZ       CHR       'CALL' CHARACTER CHANGE.
                                    *                        RETURN IS TO MAIN.
1795     FE56          0200              CPI      'V'       IF INPUT WAS 'V',
1797     CAAC17        0220              JZ       NEG       'CALL' VIDEO REVERSE.
                                    *                        RETURN IS TO MAIN.
179A     FE12          0240              CPI      'R'-40H   IF INPUT WAS CONTROL-R,
179C     CA29CA        0260              JZ       FILEI     'CALL' CASSETTE INPUT.
                                    *                        RETURN IS TO MAIN.
179F     FE13          0280              CPI      'S'-40H   IF INPUT WAS CONTROL-S,
```

CRYSTALS

THESE FREQUENCIES ONLY

Part #	Frequency	Case/Style	Price
CY1A	1.000 MHz	HC33/U	$5.95
CY2A	2.000 MHz	HC33/U	$5.95
CY3A	4.000 MHz	HC18/U	$4.95
CY7A	5.000 MHz	HC18/U	$4.95
CY12A	10.000 MHz	HC18/U	$4.95
CY14A	14.31818 MHz	HC18U	$4.95
CY19A	18.000 MHz	HC18/U	$4.95
CY22A	20.000 MHz	HC18/U	$4.95
CY30B	32.000 MHz	HC18/U	$4.95

XR-2260KB Kit $27.95 **XR-2206KA Kit $17.95**

EXAR

WAVEFORM GENERATORS

XR-205	$8.40
XR-2206CP	4.49
XR-2207CP	3.85

STEREO DECODERS

XR-1310CP	$3.20
XR-1310EP	3.20
XR-1800P	3.20
XR-2567	2.99

MISCELLANEOUS

XR-2211CP	$6.70
XR-4136	.99
XR-1468	3.85
XR-1488	5.80
XR-1489	4.80
XR-2208	5.20

TIMERS

XR-555CP	$ .99
XR-320P	1.55
XR-556CP	1.85
XR-2556CP	3.20
XR-2240CP	3.25

PHASE LOCKED LOOPS

XR-210	5.20
XR-215	6.60
XR-567CP	1.95
XR-567CT	1.70

CONNECTORS

PRINTED CIRCUIT EDGE-CARD

.156 Spacing-Tin-Double Read-Out

Bifurcated Contacts — Fits .054 to .070 P.C. Cards

15/30	PINS (Solder Eyelet)	$1.95
18/36	PINS (Solder Eyelet)	$2.49
22/44	PINS (Solder Eyelet)	$2.95
50/100 (.100 Spacing)	PINS (Solder Eyelet)	$6.95

25 PIN-D SUBMINATURE

DB25	PLUG	$3.25
DB25	SOCKET	$4.95

iMC 3½ DIGIT DVM KIT

This 0-2 VDC .05 per cent digital voltmeter features the Motorola 3½ digit DVM chip set. It has a .4" LED display and operates from a single +5V power supply. The unit is provided complete with an injection molded black plastic case complete with Bezel. An optional power supply is available which fits into the same case as the 0-2V DVM allowing 117 VAC operation.

A. 0-2V DVM with Case $49.95

B. 5V Power Supply $14.95

VECTOR WIRING PENCIL

Vector Wiring Pencil Pt 73 consists of a hand held featherweight (under one ounce) tool which is used to guide and wrap insulated wire fed off a self-contained replaceable bobbin onto component leads or terminals installed on pre-punched "P" Pattern Vectorboard. Connections between the wrapped wire and component leads, pads or terminals are made by soldering. Complete with 250 FT of red wire. **$9.95**

REPLACEMENT WIRE — BOBBINS FOR WIRING PENCIL

W36-3-A-Pkg 3	250 ft 36 AWG GREEN	$2.40
W36-3-B-Pkg 3	250 ft 36 AWG RED	$2.40
W36-3-C-Pkg 3	250 ft 36 AWG CLEAR	$2.40
W36-3-D-Pkg 3	250 ft 36 AWG BLUE	$2.40

1/16 VECTOR BOARD

	0.1" Hole Spacing Part No	P-Pattern L	P-Pattern W	Price 1	Price 2-Up
PHENOLIC	64P44 062XXXP	4.50	6.50	1.72	1.54
	169P44 02XXXP	4.50	17.00	3.69	3.32
EPOXY GLASS	64P44 062	4.50	6.50	2.07	1.86
	84P44 062	4.50	8.50	2.56	2.31
	169P44 062	4.50	17.00	5.04	4.53
	169P84 062	8.50	17.00	9.23	8.26
EPOXY GLASS COPPER CLAD	169P44 062C1	4.50	17.00	6.80	6.12

HEAT SINKS

205-CB	Beryllium Copper Heat Sink with Black Finish for TO-5	$ .25
291-.36H	Aluminum Heat Sink for TO-220 Transistors & Regulators	$ .25
680-.75A	Black Anodized Aluminum	$1.60

HEXADECIMAL ENCODER 19-KEY PAD

- 1 - 0
- ABCDEF
- Return Key
- Optional Key (Period)
- — Key

$10.95 each

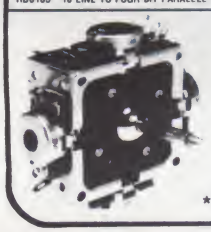
63 KEY KEYBOARD

This keyboard features 63 unencoded SPST keys unattached to any kind of P.C.B. A very solid molded plastic 13" x 4" base suits most applications.

$19.95

HD0165 16 LINE TO FOUR BIT PARALLEL KEYBOARD ENCODER $7.95

JOYSTICK



These joysticks feature four * potentiometers, that vary resistance proportional to the angle of the stick. Sturdy metal construction with plastics components only at the movable joint. Perfect for electronic games and instrumentation.

***5K Pots $6.95**

***100K Pots $7.95**

MICROPROCESSOR COMPONENTS

8080A	CPU	$19.95	8228	System Controller - Bus Driver	$10.95
8212	8 Bit Input/Output	4.95	MC6800L	8 Bit MPU	35.00
8214	Priority Interrupt Control	15.95	MC6820L	Periph. Interface Adapter	15.00
8216	Bi-Directional Bus Driver	6.95	MC6810AP1	128 x 8 Static RAM	6.00
8224	Clock Generator/Driver	10.95	MC6830L7	1024 x 8 Bit ROM	18.00

CPU'S

8080	Super 8008	24.95
8080A	Super 8008	19.95

SR'S

2504	1024 Dynamic	$ 3.95
2518	Hex 32 BIT	7.00
2519	Hex 40 BIT	4.00
2524	512 Dynamic	2.49
2525	1024 Dynamic	6.00
2527	Dual 256 BIT	3.95
2529	Dual 512 BIT	4.00
2532	Quad 80 BIT	3.95
2533	1024 Static	7.95
3341	Fifo	6.95
74LS670	16 x 4 Reg	3.95

UART'S

AY-5-1013	30K Baud	$5.95

ROM'S

2513	Char. Gen	$ 9.95
2516	Char. Gen	10.95
74S387	1024-Bit Programmable	1.95

RAM'S

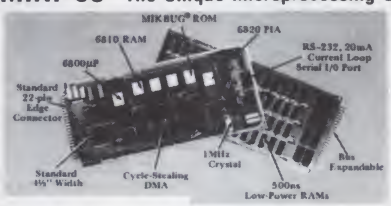
1101	256 x 1	Static	$ 1.49
2101	256 x 4	Static	5.95
2102	1024 x 1	Static	1.75
2107/5280	4096 x 1	Dynamic	4.95
2111	256 x 4	Static	6.95
7489	16 x 4	Static	2.49
8101	256 x 4	Static	6.95
8111	256 x 4	Static	6.95
8599	16 x 4	Static	3.49
91L02	1024 x 1	Static	2.25
74200	256 x 1	Static	6.95
93421	256 x 1	Static	2.95
MM5262	2K x 1	Dynamic	2 for 1.00

PROMS

1702A	2048	Famos	$ 9.95
5203	2048	Famos	14.95
82S23	32 x 8	Open C	5.00
82S123	32 x 8	Tristate	5.00
74S287	1024	Static	7.95
3601	256 x 4	Fast	3.95
2708	8K	Eprom	34.95
6301-1	1024	Tri-State Bipolar	3.49
6330-1	256	Open Collector Bipolar	2.95
6331-1	256	Tri-State Bipolar	2.95

GEMINI-68 The Unique Microprocessing System

Standard 4½" Boards - Dual 22 pin edge Connector NOT A KIT



Fully Buffered and Tristatable Address and Data Buses FULLY ASSEMBLED

STAND ALONE CPU BOARD — Has 384 bytes of RAM on board, serial I/O (RS-232 and 20 ma current loop, cycle stealing direct memory access (DMA), built in soft-ware — selectable echo-back capability
Part # SA-CPU Board **$279.95**

CPU BOARD — Same as above but only has 128 bytes of RAM on board-used with 8K RAM board listed below
Part # Gemini 68 CPU Board **$259.95**

8K RAM BOARD — Uses low power static RAMS, 500ns cycle time. 1.5 Amps Max
Part # Gemini 68 RAM Board **$269.95**

8K EPROM BOARD — Uses 5204 EPROMS by AMI or NATIONAL. Shipped with all decode and miscellaneous IC's, except the 5204 EPROMS
Part # Gemini 68 EPROM Board **$ 89.95**

Allow approximately four weeks per delivery

SPECIAL REQUESTED ITEMS

MC3061P	3.50	CD4508	6.75	82S115	25.00	334t	6.95
MC4016P (74416)	7.50	CD4515	6.50	5841	9.95	9368	3.95
MC14583	3.50	CD4520	2.70	MK50240	17.50	MC1408L7	9.95
MC14562	14.50	MCM6571	17.50	11C90	19.95	LD110/LD111	25.00 set
CD4059	9.95	MCM6574	17.50	DS0026CH	3.75	AY-5-9100	17.50 ea
CD4070	.95	MCM6575	17.50	TL308	10.50	9589J	13.95

THE BUGBOOKS— Integrated Circuits Applications & Experiments

BUGBOOK I & II - Basic concepts of TTL Logic — over 90 experiments **$17.00/set**

BUGBOOK IIa - Introduces UART — recommended for RTTY enthusiast **$5.00/book**

BUGBOOK III - Explores 8080 chip — introduces Mark 80 Microcomputer **$15.00/book**

555 TIMER APPLICATIONS SOURCEBOOK WITH EXPERIMENTS — over 100 design techniques **$6.95/book**

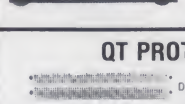
Introductory Offer — all 5 books (worth $43.95) **SPECIAL - $39.95**

CONTINENTAL SPECIALTIES

PROTO BOARD 6

$15.95

(6" long X 4" wide)



Other CS Proto Boards

PB100 - 4.5" x 6"	$ 19.95
PB101 - 5.8" x 4.5"	29.95
PB102 - 7" x 4.5"	39.95
PB103 - 9" x 6"	59.95
PB104 - 9.5" x 8"	79.95
PB203 - 9.75 x 6½ x 2¾	75.00
PB203A - 9.75 x 6½ x 2¾ (includes power supply)	120.00

Logic Monitor $84.95
for DTL, HTL, TTL or CMOS Devices

PROTO CLIPS

14 PIN	$4.50
16 PIN	4.75
24 PIN	8.50

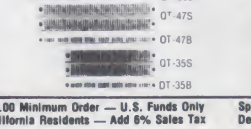
DESIGN MATES

DM1 - Circuit Designer 54.95

DM2 - Function Generator 69.95

DM3 - RC Bridge 59.95

QT PROTO STRIPS



QT type	#holes	price
QT-59S	590	12.50
QT-59B	bus strip	2.50
QT-47S	470	10.00
QT-47B	bus strip	2.25
QT-35S	350	8.50
QT-35B	bus strip	2.00
QT-18S	180	4.75
QT-12S	120	3.75
QT-8S	80	3.25
QT-7S	70	3.00

Experimentor 300	$ 9.95
Experimentor 600	$10.95

$5.00 Minimum Order — U.S. Funds Only
California Residents — Add 6% Sales Tax

Spec Sheets - 25¢ — Send 35¢ Stamp for 1977A Catalog
Dealer Discount Available — Request Pricing

NOW OPEN SATURDAYS

James ELECTRONICS

1021-A HOWARD AVE., SAN CARLOS, CA. 94070
PHONE ORDERS WELCOME — (415) 592-8097

J-1

All Advertised Prices Good Thru April

Panatronics

LOGIC ANALYZER KIT

(as shown in Feb. Popular Electronics)

Troubleshoot any type of digital system including Microcomputers.

(includes IC, power supply, PC Board, case and manual)

LA100 KIT $189.00

TV ANTENNA SWITCH

- More than 60 db isolation
- Switches 300 ohm to 300 ohm
- Switches 75 ohm to 300 ohm
- RF coax connecting cable (16ft.) with RCA phono plyes

$6.95

CB MOBiLE MONiTOR

"Hear the highway action"

- Off-on volume control
- Internal mount antenna
- 9-volt battery powered
- Portable — attaches to visor
- 23-Channel operation
- Long range reception (Battery not included)

Actual Size - 4" x 3" x 1½

$9.95

DIGITAL STOPWATCH

- Bright 6 Digit LED Display
- Times to 59 minutes 59.59 seconds
- Crystal Controlled Time Base
- Three Stopwatches in One Times Single Event — Split & Taylor
- Size 4.5" x 2.15" x .90" (4½ ounces)
- Uses 3 Penlite Cells

Kit — $39.95

Assembled — $49.95

Heavy Duty Carry Case $5.95

Stop Watch Chip Only (7205) $19.95

Timeband

A Trademark of Fairchild Camera and Instrument Corporation

DIGITAL ALARM CLOCK $16.95

- 24-Hour Alarm
- "DOZE" Button
- 100% Solid State
- Large Red Led Display (.8" high)
- AM/PM Indicator
- Seconds Display at touch of button
- SPECIFY BLACK OR IVORY

DIGITAL QUARTZ CAR CLOCK

Complete kit from mounting bracket of the injection molded case down to the three conductor power cord and all components including MM5314 clock chip. Features quartz accuracy of .01% six digit .35" high LED display and P.C. Boards. Works on any 12 volt system — motorcycles, boats, vans, motorhomes, autos and trucks.

DIMENSIONS: 4½" x 4" x 2"

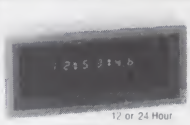
12 or 24 HOUR MODE

Kit: $29.95

Assembled: $39.95

CASE ONLY (includes hardware, mounting bracket and bezel) **$6.50**

JE700 CLOCK



12 or 24 Hour

115 VAC

The JE700 is a low cost digital clock but is a very high quality unit. The unit features a simulated walnut case with dimensions of 6" x 2½" x 1". It utilizes a MAN72 high brightness readout and the MM5314 clock chip.

$17.95

DIGITAL CLOCK KIT — 3½ INCH DIGITS

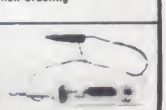
4 DIGIT KIT $49.95 **4 DIGIT ASSEMBLED $59.95**

6 DIGIT KIT $69.95 **6 DIGIT ASSEMBLED $79.95**

This clock features big 3½" high digits for viewing in offices, auditoriums, etc. Each digit is formed by 31 bright 0.2" LED's. The clock operates from 117 VAC, has either 12 or 24 hr. operation. The 6 digit version is 27" x 3½" x 1½" and the 4 digit is 18" x 3½" x 1½". Kits come complete with all components, case and transformer.

Specify 12 or 24 Hour When Ordering

JE803 PROBE



The Logic Probe is a unit which is for the most part indispensible in trouble shooting logic families TTL, DTL, RTL, CMOS. It derives the power it needs to operate directly off of the circuit under test, drawing a scant 10 mA max. It uses a MAN1 readout to indicate any of the following states by these symbols: (H), (L), (O), (PULSE), (P). The Probe can detect high frequency pulses to 45 MHz. It can't be used at MOS levels or circuit damage will result.

$9.95 Per Kit

printed circuit board

T²L 5V 1A Supply

This is a standard TTL power supply using the well known LM309K regulator IC to provide a solid 1 AMP of current at 5 volts. We try to make things easy for you by providing everything you need in one package including the hardware for only

$9.95 Per Kit

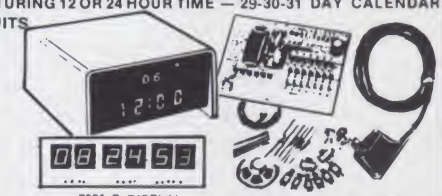
```
17A1  CAED17  0300          JZ    SAV       'CALL' SAVE SCREEN.
                      *                     RETURN IS TO MAIN.
17A4  FE0C    0320          CPI   'L'-40H   IF INPUT WAS CONTROL-L
17A6  CAF617  0340          JZ    LEV       'CALL' EXIT ROUTINE.
17A9  C3BC17  0360          JMP   MOV       ELSE IT MUST BE A TOGGLE
                      *                     KEY. 'CALL' THE MOVE
                      *                     ROUTINE. RETURN IS TO MAIN.
17AC  3AF817  0380    NEG   LDA   CHAR      GET THE CURRENT DRAWING
                      *                     CHARACTER.
17AF  EE80    0400          XRI   80H       REVERSE '7' BIT.
17B1  32F817  0420          STA   CHAR      REPLACE CURRENT DRAWING
                      *                     CHARACTER.
17B4  C9      0440          RET             EXIT SUBROUTINE.
17B5  CDC6CA  0460    CHR   CALL  KEYIN     GET ANOTHER CHARACTER FROM
                      *                     KEYBOARD.
17B8  32F817  0480          STA   CHAR      USE TO REPLACE CURRENT
                      *                     DRAWING CHARACTER.
17BB  C9      0500          RET             EXIT THE SUBROUTINE.
17BC  3AF817  0520    MOV   LDA   CHAR      GET THE DRAWING CHARACTER
17BF  2AF917  0540          LHLD  SCR       GET THE CURRENT LOCATION
                      *                     OF THE MARKER ON THE SCREEN.
17C2  77      0560          MOV   M,A       REPLACE MARKER ON SCREEN
                      *                     WITH DRAWING CHARACTER.
17C3  79      0580          MOV   A,C       RESTORE KEYBOARD INPUT
                      *                     FROM REG. C.
17C4  CD0018  0600          CALL  TOGGL     GO CHANGE VALUE IN REGISTER
                      *                     PAIR HL BASED ON TOGGLE
                      *                     VALUE IN A. HL MUST BE
                      *                     WITHIN THE AREA OF THE
                      *                     SCREEN. THE WAY THE VALUE
                      *                     IN HL IS MOVED BY A IS:
                      *                           Q  W  E
                      *                            \ | /
                      *                         A ← HL → D
                      *                            / | \
                      *                           Z  X  C
17C7  00      0620          DB    0         DONT TOGGLE UP IN LINE 0.
17C8  00      0640          DB    0         DONT TOGGLE LEFT IN POS. 0.
17C9  0F      0660          DB    15        DONT TOGGLE DOWN IN LINE 15.
17CA  3F      0680          DB    63        DONT TOGGLE RITE IN POS. 63.
17CB  C3D117  0700          JMP   $+3       TOGGL RETURNS HERE IF
                      *                     VALUE IN A IS NOT VALID.
17CE  22F917  0720          SHLD  SCR       STORE NEW CURRENT LOCATION
                      *                     OF THE MARKER ON THE SCREEN.
17D1  3600    0740          MVI   M,0       PLACE MARKER ON SCREEN.
17D3  C9      0760          RET             EXIT THE ROUTINE
17D4  2100CC  0780    CLR   LXI   H,TV      GET ADDRESS OF TV SCREEN.
17D7  E600    0800          ANI   0         CLEAR ACCUMULATOR.
17D9  D3C8    0820          OUT   TVOUT     RESET TV INTERFACE BOARD.
17DB  3ED0    0840          MVI   A,ODOH    (TV+TVL)/256 A HAS THE VALUE THAT H
                      *                     WILL HAVE WHEN HL POINTS TO
                      *                     ONE BYTE PAST SCREEN AREA.
17DD  3620    0860    LPP   MVI   M,' '     CLEAR A SCREEN POSITION.
17DF  23      0880          INX   H         GET NEXT SCREEN POSITION.
17E0  BC      0900          CMP   H         COMPARE TO ONE BYTE PAST
                      *                     SCREEN.
17E1  C2DD17  0920          JNZ   LPP       LOOP IF STILL ON SCREEN.
17E4  21C0CF  0940          LXI   H,TV+TVL-LINE  LINE GET BEGINNING ADDRESS
                      *                     OF MARKER ON SCREEN (POS. 0
                      *                     OF LINE 15).
17E7  3600    0960          MVI   M,0       PLACE MARKER THERE.
17E9  22F917  0980          SHLD  SCR       INITIALIZE CURRENT LOCATION
                      *                     OF MARKER.
17EC  C9      1000          RET             EXIT THE SUBROUTINE.
17ED  2100CC  1020    SAV   LXI   H,TV      THE BEGINNING ADDRESS TO BE
                      *                     WRITTEN TO CASSETTE IS THE
                      *                     BEGINNING OF THE TV SCREEN.
17F0  110004  1040          LXI   D,TVL     THE LENGTH IS THE SIZE
                      *                     OF THE TV SCREEN.
17F3  C37CCA  1060          JMP   FILE0     'CALL' CASSETTE OUTPUT.
                      *                     IT WILL EXIT TO WHEREVER
                      *                     SAV WOULD HAVE EXITED TO.
17F6  E1      1080    LEV   POP   H         IGNORE SUBROUTINE EXIT
                      *                     POINT.
17F7  C9      1100          RET             RETURN TO MY CALLER'S
                      *                     CALLER (PRESUMABLY THE
                      *                     MONITOR).
17F8  A0      1120    CHAR  DB    ' '+80H   USE REVERSE VIDEO SPACE AS
                      *                     DRAWING CHARACTER FIRST
                      *                     TIME THROUGH.
17F9          1140    SCR   DS    2         CONTAINS LOCATION OF
                      *                     MARKER ON SCREEN.
17FB          1160    FILE0 EQU   0CA7CH    ADDRESS OF CASSETTE OUTPUT
                      *                     ROUTINE.
17FB          1180    FILEI EQU   0CA29H    ADDRESS OF CASSETTE INPUT
                      *                     ROUTINE.
17FB          1200    TV    EQU   0CC00H    LOCATION OF TV SCREEN.
17FB          1220    TVL   EQU   400H      LENGTH OF THE TV SCREEN.
17FB          1240    LINE  EQU   64        LENGTH OF LINE ON TV SCREEN.
17FB          1260    TVOUT EQU   0C8H      ADDRESS OF SCREEN CONTROL.
                      *                     (THIS IS AN EXTREMELY
                      *                     COMPLEX FACILITY OF THE
                      *                     VDM-1 AND IT IS NOT USED
                      *                     BY THIS PROGRAM — EXCEPT
                      *                     TO BE CLEARED.)
17FB          1280    KEYIN EQU   0CAC6H    ADDRESS OF KEYBOARD
                      *                     ROUTINE.
17FB          1300    TOGGL EQU   1800H     ADDRESS OF TOGGLE ROUTINE.
```

6-DIGIT LED CLOCK CALENDAR KIT

DATE-TIME-SNOOZE ALARM & MORE... KIT 7001I

OUR TOP OF THE LINE KIT FOR THE BUILDER THAT WANTS THE BEST. A TOTAL PACKAGE, FEATURING 12 OR 24 HOUR TIME — 29-30-31 DAY CALENDAR WITH ALARM, SNOOZE AND AUX. TIMER CIRCUITS.

Will alternate time (8 seconds) and date (2 seconds) or may be wired for time or date display only, with other functions on demand. Has built-in oscillator for battery back-up. A loud 24 hour alarm with a repeatable 10 minute snooze alarm, alarm set & timer set indicators. Includes 110 VAC/60Hz power pack with cord and top quality components through-out.



COMPLETE KIT WITH YOUR CHOICE OF DIGITAL DISPLAYS

KIT - 7001B WITH 6 - .4" DIGITS $39.95
KIT - 7001C WITH 4 - .6" DIGITS & 2 - .3" DIGITS FOR SECONDS $42.95
KIT - 7001X WITH 6 - .6" DIGITS $45.95

7001 X DISPLAY

7001C DISPLAY

7001 B DISPLAY

KITS ARE COMPLETE (**LESS CABINET**) WITH PC BOARDS, POWER SUPPLY, IC & SOCKET, 16 TRANSISTORS, 9 SWITCHES AND ALL REQUIRED PARTS. ALL 7001 KITS FIT CABINET I AND ACCEPT (OPTIONAL) QUARTZ CRYSTAL TIME BASE KIT # TB-1

$39.95 ea.

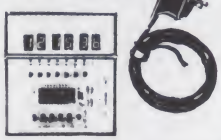
6 DIGIT LED CLOCK KIT #850-4

12/24 HR. OPERATION BIG .4" DIGITS - 50/60 HZ OPERATION.

KIT INCLUDES
- INSTRUCTIONS
- QUALITY COMPONENTS
- 50 or 60 Hz OPERATION
- 12 or 24 HR OPERATION

6-LED Readouts (FND-359 Red, com. cathode)
1-MM5314 Clock Chip (24 pin)
13-Transistors
3-Switches
6-Capacitors
5-Diodes
9-Resistors
24-Molex pins for IC socket

LARGE .4" DIGITS! ORDER KIT #850-4 AN INCREDIBLE VALUE!



$11.95 ea. QTY. 1-5
$10.95 ea. QTY. 6-11
$9.95 ea QTY. 12 OR MORE

"Kit #850-4 will furnish a complete set of clock components as listed. The only additional items required are a 7-12 VAC transformer, a circuit board and a cabinet, if desired."

PRINTED CIRCUIT BOARD FOR KIT #850-4, SCREEN PRINTED DRILLED AND SOLDER PLATED FIBERGLASS $2.95
MINI-BRITE RED LED'S (FOR COLON IN CLOCK DISPLAY) ..... Pkg. of 5-$1.00
MOLDED PLUG TRANSFORMER 115/10 VAC (WITH CORD) $2.50

NOTE: Entire Clock may be assembled on one PC Board or Board may be cut to remote display.
Kit #850-4 will fit Plexiglas Cabinet II.

MOBILE LED CLOCK

MODEL #2001

12 OR 24-HOUR OPERATION
12 VOLT AC or DC POWERED FOR FIXED OR MOBILE OPERATION.
SIX LARGE .4" DIGITS!

Approx. Size: 1¾"H x 4"W x 4½"D

- 6 JUMBO .4" RED LED'S BEHIND RED FILTER LENS WITH CHROME RIM
- SET TIME FROM FRONT VIA HIDDEN SWITCHES • 12/24-Hr. TIME FORMAT
- STYLISH CHARCOAL GRAY CASE OF MOLDED HIGH TEMP. PLASTIC
- BRIDGE POWER INPUT CIRCUITRY — TWO WIRE NO POLARITY HOOK-UP
- OPTIONAL CONNECTION TO BLANK DISPLAY (Use When Key Off In Car, Etc.)
- TOP QUALITY PC BOARDS & COMPONENTS - EXCELLENT INSTRUCTIONS
- MOUNTING BRACKET INCLUDED

KIT #2001 COMPLETE KIT (Less 9V. Battery) **29.95 EA.** 3 OR MORE **$27.95 EA.**
115 VAC Power Pack #AC-1 **$2.50 EA.**

ASSEMBLED UNITS WIRED & TESTED ORDER #2001 WT (LESS 9V. BATTERY) WIRED FOR 12 HR. OP. IF NOT SPECIFIED OTHERWISE **$39.95 EA.** 3 OR MORE **$37.95 EA.** Assembled Units May Be Mixed With Kits for Qty. Price

JUMBO DIGIT CLOCK KIT
A complete Kit (less Cabinet) featuring: six .5" digits, MM5314 IC, 12/24 Hr. time, 50/60 HZ., Plug-Transformer, Line Cord, Switches, and all Parts. (Ideal Fit in Cabinet II)
Kit # 5314-5 .... **$19.95 ea 2/$38.**

JUMBO DIGIT CONVERSION KIT $9.95 ea.
Convert small digit LED clock to large .5" displays. Kit includes 6 - .5" LED's, Multiplex PC Board & easy hook-up info.
Kit # JD-1CC For common Cathode
Kit # JD-1CA for common Anode

PRINTED CIRCUIT BOARDS for CT-7001 Kits sold separately with assembly info. PC Boards are drilled Fiberglass, solder plated and screened with component layout.
Specify for 7001 *B, C or D - $7.95*

TELEPHONE FORMAT KEYBOARD
BY Chomerics
$4.95 2-1/4"x3" 5/32" thick
6/$28. # EF-21360

25 AMP BRIDGE
$1.95 ea.
3/$5.00
100 PIV

CABINET I
3"H, 6¼"W, 5½"D
CABINET II
2½"H, 5"W, 4"D
ANY SIZE/COLOR $6.50 ea 2/$12.

PLEXIGLAS CABINETS
Great for Clocks or any LED Digital project. Clear-Red Chassis serves as Bezel to increase contrast of digital displays.
Black, White or Clear Cover

RED OR GREY PLEXIGLAS FOR DIGITAL BEZELS
3"x6"x1/8" 95¢ ea. 4/$3

7-SEG LED

COMMON CATHODE

	COLOR	HT.	DEC PT.	PR	EA
FND-359	RED	.4"	RHDP	$	.95
FND-503	RED	.5"	RHDP	$	1.35
DL-750	RED	.6"	LHDP	$	2.95
XAN-654	GREEN	6"	NDP	$	1.95
XAN-664	RED	6"	NDP	$	1.95

COMMON ANODE

DL-747	RED	.6"	LHDP	$1.95
XAN-72	RED	.3"	LHDP	$1.25
MAN-72	RED	.3"	LHDP	$1.25
XAN-81	YELLOW	.3"	RHDP	$1.75
XAN-351	GREEN	.3"	RHDP	$1.50
XAN-361	RED	.3"	RHDP	$1.50
XAN-362	RED	.3"	LHDP	$1.50
XAN-662	RED	.6"	NDP	$1.95
XAN-692	RED	.6"	NDP	$1.95

SET OF 6 FND-359 WITH MULTIPLEX PC BOARD — 6.95

Fairchild Super Digit FND-359
.4" Char. Ht. 7 segment LED RED Com. Cath. Direct pin replacement for popular FND-70.

95¢ ea, 10/$8.50 100/$79.00

MOLEX PINS
Form Inexpensive Sockets

100 for $1.25
Reel of 1000 - $8.50

SCHOTTKY TTL

74S00	$ .35
74S01	.40
74S04	.55
74S05	.60
74S09	.55
74S10	.40
74S20	.50
74S22	.45
74S40	.45
74S50	.45
74S51	.55
74S60	.85
74S64	.55
74S74	.85
74S75	1.75
74S78	1.50
74S86	.95
74S107	.95
74S112	.95
74S113	1.40
74S114	.95
74S133	.75
74S134	.75
74S138	1.75
74S139	1.50
74S151	1.95
74S153	1.95
74S155	1.95
74S156	1.95
74S157	1.80
74S158	2.50
74S174	2.50
74S175	2.50
74S181	2.95
74S182	1.95
74S251	2.75

DIGITAL CLOCK IC's

MM5312	$ 4.95
MM5314	3.95
MM5375 AB	3.95
CT-7001	7.95
CT-7002	13.95
50380	3.95
MM5369	2.50

LED DRIVERS

7447	$ .95
7448	.95
75491	.65
75492	.65

VOLTAGE REGULATORS

LM 309 H	TO-5	$ .95
LM 309 K	TO-3	1.25
7805	TAB	.95
7812	TAB	1.25
7812	TO-3	1.50
7815	TO-3	1.25
7815	TAB	1.25
78L 15	TO-5	.75
7824	TO-3	1.25
723	DIP	.75
723	TO-5	.75

PROM

1702 E Prom $8.95
5203 E Prom $8.95

SWITCHES

ROCKER SPDT 6/$1
MINI-SLIDE SPDT 5/$1
REG. SLIDE DPDT 6/$1
PUSH BUTTON N.O. 3/$1
MINI TOGGLE SPDT 1.30
DPDT 1.50

IC SOCKETS

PINS	1-24	25	100
8	$ .25	$ .22	$ .20
14	.25	.22	.20
16	.28	.25	.23
18	.31	.28	.26
24	.50	.45	.40
28	.60	.55	.50
40	.75	.70	.65

XTAL

3.579545 MHZ. $1.95

EXAR

XR 2556 $ 1.75
XR 2567 $ 1.95

TRANSISTORS

2N2222	TO-18	5/$1.00
2N2554	TO-5	2/$1.00
2N2712	TO-98	5/$1.00
2N3415	TO-98	5/$1.00
2N3704	TO-92	5/$1.00
2N4400	TO-92	5/$1.00
2N4125	TO-92	5/$1.00
2N4249	TO-92	5/$1.00
2N4437	TO-92	5/$1.00
2N6027	PUT	2/$1.00
2N5457	N J-Fet	2/$1.00

DIODES

IN 4002	1A, 100 PIV	12/$1.00
IN 4005	1A, 600 PIV	11/$1.00
IN 4007	1A, 1000 PIV	10/$1.00
RECTIFIER	2.5A, 1000 PIV	4/$1.00
IN 914	SIL. SIGNAL	20/$1.00
IN 4148	SIL. SIGNAL	20/$1.00
DYAC	28V.	4/$1.00

LINEAR

555 TIMER 2/$1.00
556 DUAL TIMER .95
565 PLL .95
566 FUNCTION GEN. 1.75
567 TONE DECODER 1.75

TRANSISTOR SOCKET
TO-5/18 GOLD PINS
5/$1.00

NYLON WIRE TIES
8" TIE-WRAP 100/$1.95
4" TIE-WRAP 100/$1.75

MOLEX PINS
REEL OF 1000 $ 8.50
STRIP OF 100 1.25

PLUG TRANSFORMERS
12 VAC at 150 MA $ 2.50
12 VAC at 500 MA 3.50
7VAC at 1.75 VA $3.50

OP AMPS
3/$1.00
301 TO-5
709 TO-5
741 DIP
741 M-DIP
741 TO-5
748 DIP

DISCRETE LED's
JUMBO RED
10 FOR $1.00
100 FOR $9.50

PC TRIM POTS
25K 6/$1.00
4.7K 6/$1.00

SPECTROL
10K 10 TURN
95c
4/$3.00

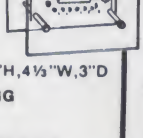
PRESCALE
11C90DC $15.95
95H90 9.95

MEMORY
450ns Fairchild 1K RAM low, low, low power-2102L-1PC 25.99 100 199 200 or more
$1.95 ea
$1.75 ea
$1.60 ea
$1.45 ea

60 HZ.
XTAL TIME BASE
Will enable Digital Clock Kits or Clock-Calendar Kits to operate from 12V DC.
1"x2" PC Board
Power Req: 5-15V (2.5 MA. TYP.)
Easy 3 wire hookup
Accuracy: ± 2PPM
#TB-1 (Adjustable)
Complete Kit **$4.95** ea.
Wir & Cal $9.95

SEE THE WORKS Clock Kit Clear Plexiglas Stand
Kit #850-4 CP
- 6 Big .4" digits
- 12 or 24 hr. time
- 3 set switches (back)
- Plug transformer
- all parts included

Plexiglas is Pre-cut & drilled



Size: 6"H, 4½"W, 3"D

A SUPER LOOKING CLOCK!
$23.50 ea. 2/$45.

CPU NS8080AD
Micro Processor Chip
$19.95 ea. Prime National LSI
40 Pin socket $.50 with each 8080A!

master charge
BANKAMERICARD

OPTOELECTRONICS, INC.
BOX 219 • HOLLYWOOD, FLA. 33022 • (305) 921 2056

ORDER BY PHONE OR MAIL. COD ORDERS WELCOME. ($1.00 CHG.) Orders Under $15 Add $1.00 Handling. Fla. Res. Please Add 4% Sales Tax. All Prepaid Orders Sent Postpaid Within Continental-USA.
OTHERS ADD 5%, 10% AIR MAIL. O-3

ADDRESS	MACHINE CODE	SEQ. NUM.	LABEL	OP-CODE	OPERANDS	COMMENTS
1800	22CE18	0020	TOGGL	SHLD	OLDH	SAVE THE CURRENT SCREEN LOCATION. WHEN ENTERED, THE HL PAIR MUST CONTAIN AN ADDRESS WITHIN THE BOUNDARIES OF THE TV SCREEN. THIS VALUE IS CHANGED (TOGGLED) BY A VALID VALUE IN THE ACCUMULATOR. THESE VALUES AND THEIR AFFECTS: Q W E A ← HL → D Z X C IF ONE OF THESE VALUES IS NOT FOUND IN THE A REG., THEN TOGGL RETURNS TO THE FOURTH BYTE BEYOND THE STANDARD RETURN POINT. IF THE VALUE IS VALID, IT RETURNS TO THE SEVENTH. THE FOUR BYTES AT THE STANDARD RETURN POINT CONTAIN TOGGLE LIMITING CONTROL INFORMATION.
1803	E1	0040		POP	H	PLACE RETURN ADDRESS IN HL.
1804	D5	0060		PUSH	D	SAVE
1805	C5	0080		PUSH	B	ALL
1806	F5	0100		PUSH	PSW	REGISTERS.
1807	5E	0120		MOV	E,M	PLACE FIRST BYTE AT RETURN POINT IN E. THIS VALUE IS THE LINE (0-15) ABOVE WHICH TOGGLING WILL FAIL.
1808	23	0140		INX	H	BYPASS THAT BYTE.
1809	56	0160		MOV	D,M	PLACE SECOND BYTE AT RETURN POINT IN D. THIS VALUE IS THE COLUMN (0-63) TO THE LEFT OF WHICH TOGGLING WILL FAIL.
180A	23	0180		INX	H	BYPASS THAT BYTE.
180B	4E	0200		MOV	C,M	PLACE THIRD BYTE AT RETURN POINT IN C. THIS VALUE IS THE LINE BELOW WHICH TOGGLING WILL FAIL.
180C	23	0220		INX	H	BYPASS THAT BYTE.
180D	46	0240		MOV	B,M	PLACE FOURTH BYTE AT RETURN POINT IN B. THIS VALUE IS THE COLUMN TO THE RIGHT OF WHICH TOGGLING WILL FAIL.
180E	23	0260		INX	H	BYPASS FOURTH BYTE.
180F	E5	0280		PUSH	H	PLACE THIS VALUE ON STACK AS NEW RETURN ADDRESS.
1810	2ACE18	0300		LHLD	OLDH	RELOAD SCREEN ADDRESS.
1813	FE57	0320		CPI	'W'	TEST EACH TOGGLE KEY
1815	CA4418	0340		JZ	A1	AND GO TO APPROPRIATE
1818	FE45	0360		CPI	'E'	TOGGLING ROUTINE:
181A	CA4D18	0380		JZ	A2	A8 A1 A2
181D	FE44	0400		CPI	'D'	A7 ← * → A3
181F	CA5918	0420		JZ	A3	A6 A5 A4
1822	FE43	0440		CPI	'C'	
1824	CA6218	0460		JZ	A4	
1827	FE58	0480		CPI	'X'	
1829	CA6E18	0500		JZ	A5	
182C	FE5A	0520		CPI	'Z'	
182E	CA7718	0540		JZ	A6	
1831	FE41	0560		CPI	'A'	
1833	CA8318	0580		JZ	A7	
1836	FE51	0600		CPI	'Q'	
1838	CA8C18	0620		JZ	A8	
183B	E1	0640	FAIL	POP	H	CONTROL FALLS THROUGH TO HERE IF A DIDN'T CONTAIN A TOGGLE CHARACTER.
183C	F1	0660		POP	PSW	POPPED RETURN PT. NOW
183D	C1	0680		POP	B	RESTORE
183E	D1	0700		POP	D	REGISTERS.
183F	E5	0720		PUSH	H	PUT RETURN POINT ON STACK.
1840	2ACE18	0740		LHLD	OLDH	RESTORE HL TO ORIGINAL.
1843	C9	0760		RET		EXIT THE SUBROUTINE.
1844	CDA518	0780	A1	CALL	TOP	GO SEE IF GOING UP IS OK. (THESE ROUTINES RETURN ONLY IF MOVING IS PERMITTED.)
1847	11C0FF	0800		LXI	D,-LINE	SET INCREMENT TO UP ONE LINE.
184A	C39518	0820		JMP	ADT	GO TO ADD IT ROUTINE.
184D	CDA518	0840	A2	CALL	TOP	GO SEE IF GOING UP IS OK.
1850	CDC518	0860		CALL	RITE	GO SEE IF GOING TO RIGHT IS OK.
1853	11C1FF	0880		LXI	D,1-LINE	SET INCREMENT TO UP ONE LINE, RIGHT ONE POSITION.
1856	C39518	0900		JMP	ADT	GO TO ADD IT ROUTINE.
1859	CDC518	0920	A3	CALL	RITE	GO SEE IF GOING TO RIGHT IS OK.
185C	110100	0940		LXI	D,1	SET INCREMENT TO RIGHT ONE POS.
185F	C39518	0960		JMP	ADT	GO TO ADD IT ROUTINE.

DIODES/ZENERS

1N914	100v	10mA	.05
1N4004	400v	1A	.08
1N4005	600v	1A	.08
1N4007	1000v	1A	.15
1N4148	75v	10mA	.03
1N753A	6.2v	z	.25
1N758A	10v	z	.25
1N759A	12v	z	.25
1N4733	5.1v	z	.25
1N5243	13v	z	.25
1N5244B	14v	z	.25
1N5245B	15v	z	.25

SOCKETS/BRIDGES

8-pin	pcb	.25	ww	.45
14-pin	pcb	.25	ww	.40
16-pin	pcb	.25	ww	.40
18-pin	pcb	.25	ww	.75
22-pin	pcb	.45		
24-pin	pcb	.35	ww	1.25
28-pin	pcb	.35	ww	1.45
40-pin	pcb	.50	ww	1.95

Molex pins	.01	To-3 Sockets	.25
2 Amp Bridge		100-prv	1.20
25 Amp Bridge		200-prv	2.50

TRANSISTORS, LEDS, etc.

2N2222	NPN			.10
2N2907	PNP			.15
2N3740	PNP	1A	60v	.25
2N3906	PNP			.10
2N3055	NPN	15A	60v	.50
T1P125	PNP	*Darlington*		.35

LED Green, Red, Clear		.15
D.L.747	7 seg 5/8" high com-anode	1.95
XAN72	7 seg com-anode	1.50
FND 359	Red 7 seg com-cathode	1.25

C MOS

4000	.20
4001	.20
4002	.25
4004	4.95
4006	1.20
4007	.40
4008	1.20
4009	.25
4010	.45
4011	.20
4012	.20
4013	.40
4014	1.10
4015	.95
4016	.35
4017	1.10
4018	1.10
4019	.70
4020	.85
4021	1.35
4022	1.15
4023	.25
4024	.75
4025	.35
4026	1.95
4027	.50
4028	.95
4030	.45
4033	1.95
4034	2.45
4035	1.25
4040	1.35
4041	.69
4042	.95
4043	1.25
4044	.95
4046	1.50
4049	.80
4050	.70
4066	1.35
4069	.40
4071	.35
4082	.45

— T T L —

7400	.15
7401	.15
7402	.20
7403	.25
7404	.15
7405	.25
7406	.35
7407	.55
7408	.25
7409	.15
7410	.15
7411	.25
7412	.30
7413	.65
7414	1.10
7416	.25
7417	.50
7420	.15
7426	.40
7427	.45
7430	.15
7432	.45
7437	.45
7438	.35
7440	.25
7441	1.15
7442	.65
7443	.95
7444	.55
7445	.95
7446	.95
7447	.95
7448	1.20
7450	.25
7451	.25
7453	.25
7454	.25
7460	.40
7470	.45
7472	.45
7473	.35
7474	.40
7475	.45
7476	.20
7480	.65
7481	.99
7483	1.00
7485	1.05
7486	.40
7489	2.50
7490	.55
7491	1.15
7492	.95
7493	.45
7494	1.25
7495	.85
7496	.95
74100	1.85
74107	.45
74121	.40
74122	.55
74123	.55
74125	.45
74132	1.35
74141	1.30
74150	1.00
74151	.95
74153	.95
74154	.75
74156	1.15
74157	.75
74161	1.25
74163	1.25
74164	.95
74165	1.50
74166	1.35
74175	.95
74176	1.25
74180	.85
74181	3.25
74182	.95
74190	1.75
74192	1.65
74193	.85
74194	1.45
74195	.95
74196	1.50
74197	1.25
74198	2.35
74367	.85
75108A	.35
75110	.35
75491	.50
75492	.50
74H00	.25
74H01	.25
74H04	.25
74H05	.25
74H11	.25
74H15	.30
74H20	.30
74H22	.40
74H30	.25
74H40	.25
74H51	.25
74H52	.15
74H53J	.25
74H55	.25
74H72	.55
74H101	.75
74H103	.75
74H106	.95
74L00	.35
74L02	.35
74L03	.30
74L10	.35
74L30	.45
74L47	1.95
74L55	.65
74L72	.45
74L75	.55
74S00	.55
74S02	.55
74S03	.50
74S04	.45
74S05	.45
74S08	.45
74S10	.45
74S11	.45
74S20	.50
74S40	.30
74S50	.35
74S51	.45
74S64	.30
74S74	.50
74S112	1.50
74S133	.45
74S140	.75
74S151A	.45
74S153	.45
74S158	.45
74S194	1.50
74S257 (8123)	.25
74LS00	.45
74LS01	.45
74LS02	.45
74LS04	.55
74LS08	.45
74LS09	.45
74LS10	.45
74LS11	.45
74LS20	.50
74LS21	.25
74LS22	.25
74LS32	.55
74LS37	.40
74LS40	.55
74LS42	1.75
74LS74	.95
74LS90	1.30
74LS93	1.00
74LS107	.95
74LS153	1.20
74LS157	.85
74LS164	1.90
74LS367	.85
74LS368	.70

9000 SERIES

9301	1.00
9309	.45
9322	1.10
9602	1.50

MEMORY, CLOCKS

74S188 (8223)	3.00
MM1702A	9.95
MM5314	3.50
MM5316	3.95
2102-1	1.75
TR 1602A	6.95
TMS6011NC	6.95
8080AD	19.50
8T23	1.50
8T24	2.00
2107B-4	5.95

LINEARS, REGULATORS, etc.

MCT2	.95
8038	3.95
LM201AH	.75
LM301AH	.25
LM308AH	1.00
LM309H	.65
LM309K	.90
LM310	1.15
LM311D (Mini)	.95
LM320K5	1.65
LM320K12	1.65
LM320T12	1.65
LM320T15	1.65
LM339	.95
7805 (340T-5)	1.00
LM340T-12	1.25
LM340T-15	1.25
LM340T-18	1.65
LM340T-24	1.25
LM340K-12	2.15
LM340K-15	1.65
LM340K-18	1.65
LM340K-24	1.25
LM373	2.95
LM380	.95
LM709 (8, 14 PIN)	.30
LM711	.45
LM723	.45
LM725	1.95
LM739	1.50
LM741 8-14	.25
LM747	1.10
LM1307	1.25
LM1458	.95
LM3900	.65
LM75451	.65
NE555	.50
NE556	1.10
NE565	.95
NE566	1.75
NE567	1.35
SN72720	.35
SN72820	.35

```
1862  CDB918  0980  A4     CALL  BOT     GO SEE IF GOING DOWN IS OK.
1865  CDC518  1000        CALL  RITE    GO SEE IF GOING RIGHT IS OK.
1868  114100  1020        LXI   D,LINE+1 SET INCREMENT TO DOWN ONE LINE,
                                         RIGHT ONE POSITION.
                *
186B  C39518  1040        JMP   ADT     GO TO ADD IT ROUTINE.
186E  CDB918  1060  A5    CALL  BOT     GO SEE IF GOING DOWN IS OK.
1871  114000  1080        LXI   D,LINE  SET INCREMENT TO DOWN ONE LINE.
1874  C39518  1100        JMP   ADT     GO TO ADD IT ROUTINE.
1877  CDB118  1120  A6    CALL  LEFT    GO SEE IF GOING LEFT IS OK.
187A  CDB918  1140        CALL  BOT     GO SEE IF GOING DOWN IS OK.
187D  113F00  1160        LXI   D,LINE-1 INCREMENT IS LEFT ONE AND DOWN ONE.
1880  C39518  1180        JMP   ADT     GO TO ADD IT ROUTINE.
1883  CDB118  1200  A7    CALL  LEFT    GO SEE IF GOING LEFT IS OK.
1886  11FFFF  1220        LXI   D,-1    SET INCREMENT TO LEFT ONE POSITION.
1889  C39518  1240        JMP   ADT     GO TO ADD IT ROUTINE.
188C  CDB118  1260  A8    CALL  LEFT    GO SEE IF GOING LEFT IS OK.
188F  CDA518  1280        CALL  TOP     GO SEE IF GOING UP IS OK.
1892  11BFFF  1300        LXI   D,-LINE-1 SET INCREMENT TO UP ONE LINE, LEFT
                                         ONE POSITION.
                *
1895  19      1320  ADT   DAD   D       ADD INCREMENT TO CURRENT LOCATION.
1896  22CE18  1340        SHLD  OLDH    CHANGE OLD POSITION (MAKE IT NEW).
1899  E1      1360  OUT   POP   H       NORMAL OUTPUT ROUTINE: GET RETURN PT.
189A  23      1380        INX   H       ADD
189B  23      1400        INX   H             THREE TO IT:
189C  23      1420        INX   H                   SUCCESSFUL RETURN PT.
189D  F1      1440        POP   PSW     RESTORE
189E  C1      1460        POP   B             THE
189F  D1      1480        POP   D                   REGISTERS.
18A0  E5      1500        PUSH  H       PUT RETURN PT. ON STACK FOR RETURN.
18A1  2ACE18  1520        LHLD  OLDH    GET LOCATION (NEW OR OLD).
18A4  C9      1540        RET           EXIT FROM SUBROUTINE.
18A5  E5      1560  TOP   PUSH  H       SAVE CURRENT LOCATION.
18A6  29      1580        DAD   H       DOUBLING HL TWICE IS THE SAME AS
18A7  29      1600        DAD   H       SHIFTING IT 2 BITS TO LEFT. THIS HAS
                *                       THE AFFECT OF PLACING THE SCREEN LINE
                *                       NUMBER IN THE BOTTOM FOUR BITS OF H.
18A8  7C      1620        MOV   A,H     PLACE CURRENT LINE NUMBER IN A AND
18A9  E60F    1640        ANI   OFH       TURN OFF EXTRANEOUS BITS IN TOP OF A.
18AB  BB      1660        CMP   E       SEE IF AT TOP LINE LIMIT.
18AC  E1      1680        OP
18AC  E1      1680        POP   H       RESTORE H IN ANY CASE.
18AD  CACA18  1700        JZ    STAY    GO TO DONT MOVE IF AT LIMIT.
18B0  C9      1720        RET           OTHERWISE RETURN.
18B1  7D      1740  LEFT  MOV   A,L     PLACE CURRENT POSITION IN A AND
18B2  E63F    1760        ANI   3FH       TURN OFF EXTRANEOUS BITS IN TOP OF A.
18B4  BA      1780        CMP   D       SEE IF AT LEFT POSITION LIMIT.
18B5  CACA18  1800        JZ    STAY    GO TO DONT MOVE IF AT LIMIT.
18B8  C9      1820        RET           OTHERWISE RETURN.
18B9  E5      1840  BOT   PUSH  H       SAVE CURRENT LOCATION.
18BA  29      1860        DAD   H       CHANGE          XXXXXXLL  LLPPPPPP
18BB  29      1880        DAD   H         INTO          XXXXLLLL  PPPPPP00.
18BC  7C      1900        MOV   A,H     PLACE LINE NUMBER IN A AND
18BD  E60F    1920        ANI   OFH       TURN OFF EXTRANEOUS BITS IN TOP OF A.
18BF  B9      1940        CMP   C       SEE IF AT BOTTOM LINE LIMIT.
18C0  E1      1960        POP   H       RESTORE H IN ANY CASE.
18C1  CACA18  1980        JZ    STAY    GO TO DONT MOVE IF AT LIMIT.
18C4  C9      2000        RET           OTHERWISE RETURN.
18C5  7D      2020  RITE  MOV   A,L     PLACE POSITION (LLPPPPPP) IN A AND
18C6  E63F    2040        ANI   3FH       CONVERT TO 00PPPPPP.
18C8  B8      2060        CMP   B       SEE IF AT RIGHT POSITION LIMIT.
18C9  C0      2080        RNZ           RETURN IF ITS O.K. TO MOVE.
18CA  E1      2100  STAY  POP   H       DONT USE SUBROUTINE RETURN PT.
18CB  C39918  2120        JMP   OUT     GO EXIT TOGGL (USING ORIGINAL OLDH).
18CE          2140  OLDH  DS    2       CURRENT POSITION HOLD AREA.
18D0          2160  LINE  EQU   64      SIZE OF VDM-1 LINE.
```

- Accuracy ±0.05% of Reading ±1 Count
- Two Voltage Ranges 1.999 V and 199.9 mV
- Up to 25 Conversions/s
- $Z_{in} > 1000$ M ohm
- Auto-Polarity and Auto-Zero
- Single-Positive Voltage Reference
- Standard B Series CMOS Outputs—Drives One Low Power Schottky Load
- Uses On-Chip System Clock, or External Clock
- Low Power Consumption: 8.0 mW typical @ ±5.0 V
- Wide Supply Range: e.g. ±4.5 V to ±8.0 V

MC14433 SINGLE CHIP 3½ DIGIT A/D

Single chip combines linear and CMOS digital to bring you the simplest yet DVM approach. Requiring only 4 external passive parts, this subsystem gives you: Auto polarity, auto zero, single voltage reference, 8 mW operation, overrange, underrange signals, 25 conversions per second and .05% ± 1 count accuracy! 100 uV resolution. 24 Pin DIP.

MC14433P.....with specs.....$19.55

MC14412 UNIVERSAL MODEM CHIP

MC14412 contains a complete FSK modulator and de-modulator compatible with foreign and USA communications. (0-600 BPS)

FEATURES:

- .On chip crystal oscillator
- .Echo suppressor disable tone generator
- .Originate and answer modes
- .Simplex, half-duplex, and full duplex operation
- .On chip sine wave
- .Modem self test mode
- .Selectable data rates: 0-200, 0-300, 0-600
- .Single supply
  - VDD=4.75 to 15VDC - FL suffix
  - VDD=4.75 to 6 VDC - VL suffix

TYPICAL APPLICATIONS:

- .Stand alone - low speed modems
- .Built - in low speed modems
- .Remote terminals, accoustic couplers

MC14412FL..... $28.99
MC14412VL..... $21.74
6 pages of data..... .60

Crystal for the above.....$4.95

MC14411 BIT RATE GENERATOR.

Single chip for generating selectable frequencies for equipment in data communications such as TTY, printers, CRT s or microprocessors. Generates 14 different standard bit rates which are multiplied under external control to 1X, 8X, 16X or 64X initial value. Operates from single +5 volt supply. MC14411..... $11.98
4 pages of data..... .40
Crystal for the above..... $4.95

74C922 HEXADECIMAL KEYBOARD ENCODER

CMOS key encoders provide all the necessary logic to fully encode an array of SPST switches. On chip oscillator for scan or use with external clock source. Output is binary and is low power TTL compatible. Keybounce elimination with single capacitor. Memory holds last key depressed. Operates on 3 to 15 volts. 2-key rollover.

MM74C922N.....16 key encoder.....$6.35
Specs for MM74C922N..... .60

MM5865 UNIVERSAL TIMER I.C.

A truly universal timer can be used for a stopwatch, kitchen timer, oven timer, event timer/counter, rally timer----. 7 functions, two counters, internal comparators, on chip oscillator. Memory for rally with total elapsed time. Can be cascaded---selectable resolution--count up or down ---selectable modulo for time or event count. Operate on 7 to 20 volts at about 7mA.

MM5865N..... $8.75
Specs and applications..... 60c

MIL RANGE 5V REGULATOR.

Want a little better performance from your 5V supply? LM140K in TO-3 from contract cancellation brings you a superior part at the price of plastic regulator.

LM140K.....$1.95

LM1889 TV VIDEO MODULATOR

The LM1889 is designed to interface audio, color difference, and luminance signals to the antenna terminals of a TV receiver. It consists of a sound subcarrier oscillator, chroma subcarrier oscillator, quadrature chroma modulators, and R.F. oscillators and modulators for two low-VHF channels. The LM1889 allows video information from VTR's, games, test equipment, or similar sources to be displayed on black and white or color TV receivers.

LM1889 with 16 pages of data $9.95, data only, $1.00

AMP L'ANNY Says

THIS MONTH I'M ONE YEAR OLD, GOIN' ON TWO. THE GUYS AND GIRLS AT TRI-TEK ARE WORKING HARD TO BRING YOU THE BEST IN PARTS AND SERVICE. WATCH OUR ADS AND FLYERS FOR SOME FULLY GROWED BARGAINS .....

COS-MAC'S BACK!

CDP1802 COSMAC IS BACK IN STOCK !!!!!!!!!
The famous and very popular 1802 is here again after a long dry spell. This is the little 8 bit CMOS microprocessor which sald out on our first offering.
Special AMP'L ANNY birthday price of $29.95 for the month of APRIL only!!

CDP1802.....$29.95
CDP1852..... Parallel adapter..... $15.95
CDP1821SCD.....$24.62
CDP1822SCD.....$25.80
CDP1824CD.....$13.35

8 Element DIP Switch with big handles. Fits 16 pin DIP sockets. Right for those on board programs.

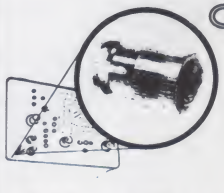
DIS-MX08.....$1.79

DATA BOOKS BY NATIONAL SEMICONDUCTOR

DIGITAL. Covers TTL, DTL, Tri-State, etc.....$3.95
LINEAR. Covers amplifiers, pre-amps, op-amps, op-amps... $4.25
LINEAR APPLICATIONS VOLUME I. Dozens of application notes and technical briefs covering the use of op-amps, regulators, phase locked loops and audio amps..... $3.25
LINEAR APPLICATIONS VOLUME II. Takes up where Volume I left you--All the latest linear devices. Along with Vol I you have a great source of application data on the most widely used devices as well as new types just appearing.....$3.25
CMOS Gates, Flip Flops, registers, etc.....$3.00
VOLTAGE REGULATORS. A must for anyone making a power supply. Complete theory including transformers, filters, heat sinks, regulators etc..... $3.00
MEMORY. Info on MOS and Bipolar memories, RAMS ROMS, PROMS and decoder/encoders..... $3.95
INTERFACE. Covers peripheral drivers, level translators, line driver/receivers, memory and clock drivers, sense amps, display driver and opto-couplers..... $3.95
SPECIAL FUNCTIONS DATA BOOK. Contains detailed information for specifying and applying special amplifiers, buffers, clock drivers, analog switches and D/A-A/D converter products..... $3.25
AUDIO HANDBOOK Contains detailed discussions, including complete design particulars, covering many areas of audio with real world design examples... $3.25

----------SPECIAL----------

DATA BOOKSHELF. Buy all ten of the National Data books at one time and save $5.10!!!.....$30.00
(All books shipped ppd in US only. Foreign orders please add shipping for 1.5 lbs per book)

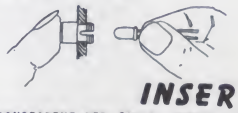


CLIPLITE™

COMBINATION LENS AND MOUNTING DEVICE FOR T 1¾ LED

REQUIRES NO TOOLS

SNAP CLIPLITE



INSERT LED

AVAILABLE IN TRANSPARENT RED · GREEN · AMBER · CLEAR & YELLOW

CLIPLITE
Combination lens and mounting device for T 1-3/4 LED. The CLIPLITE combines the benefits of the present LED display panel mounting methods and eliminates their deficiencies. Requires no special tools and installs in 6 seconds in .250" hole. Simple two-step installation. Just snap CLIPLITE, insert LED. Available in transparent red, green, amber, clear and yellow. Specify colors, any mix.

5/$1.00, 10/$1.90, 20/$3.50, 50/$7.50, 100/$13.50

30V DIAC. Tiny glass diode for triggering SCR's and triacs. Improve your firing circuit for more reliable and repeatable operations. At a fraction of the O.E.M. price!! DIA-0030.....10/$1

MINIATURE POWER SUPPLY
5V, .25A, P.C. mounting module supplies logic voltage from 105-125VAC input, 50-400HZ. Ultra-stable and noise free with external trim capability. Made by PMC. (Model MM-5BL).....$15.95

LINEAR CONTROL DATA BOOK
T.I. linear products data and applications. Price includes shipping charge, U.S. only..... $4.25
Outside U.S., add postage for 2 lbs.

NEW BOOK FROM NATIONAL
MOS LSI. Giant data book filled with spec and apps on large scale MOS circuits from National Semiconductor Corporation. Price includes shipping in U.S. only...$4.25
Outside U.S., add postage for 2 lbs.

tri-tek, inc.
6522 north 43rd avenue,
glendale, arizona 85301
phone 602 - 931-6949

We pay surface shipping on all orders over $10 US, $15 foreign in US funds. Please add extra for first class or air mail. Excess will be refunded. Orders under $10, add $1 handling. Please add 50¢ insurance. Master charge and Bank America cards welcome, ($20 minimum). Telephone orders may be placed 10AM to 5:30PM daily, Mon thru Fri. Call 602-931-4528. Check reader service card or send stamp for our latest flyers packed with new and surplus electronic components.

RONDURE COMPANY

the computer room

Where We Ship from Inventory the Same Day Your Order Arrives*

A SELECTRIC TERMINAL COMPLETE WITH RS-232/C INTERFACE AND CERTIFIED FOR MAINTENANCE BY A NATIONAL SERVICE COMPANY.

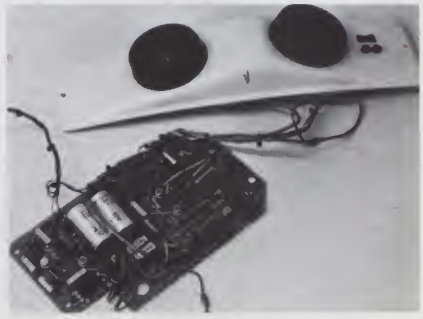
$895.00

*Maintenance limited to cities in which service now offered. Shipped the same day as certified check or money order arrives. When regular checks accompany order, equipment is shipped when regular check clears.

Specifications

- ☐ Size: 21" wide x 21" deep x 8" high.
- ☐ Power Input: 115 Volt, 60 Hz.
- ☐ Mounting: Tabletop.
- ☐ Interface: RS232C. (BCD code)
- ☐ Weight: 54 lbs.
- • Documentation $20 with unit.
- • ASCII conversion available.
- • Shipping – collect

CANNON 25 PIN RS232 PLUGS $1.50 each Min. order 10
MODEL 33 KSR TELETYPES w/coupler $595.00
MODEL 38 ASR ... NEW $1095.00
12" MOTOROLA MONITOR (20 MHz Bandwidth, 800 lines) – NEW $160.00
LINE PRINTERS – USED – 150 1pm $895
LINE PRINTERS – USED – 480 1pm $995.00



SMALL ITEMS

TERMS: Check or Money Order. Add $2.00 shipping and handling. All others shipping packaging and shipping collect.

ACCOUSTICAL MODEMS – ORIGINATE ONLY
USED – UNTESTED
Physically fit into Model 33 Teletype. Manufactured by Paragon partial documentation 2 for $25

ACCOUSTICAL MODEMS – ORIGINATE ONLY
USED – UNTESTED
IN WOOD ENCLOSURE
$20.00 ea. 2 for $35

RONDURE COMPANY
NEW ADDRESS
2522 BUTLER
DALLAS, TEXAS 75235
Phone: (214) 630-4621

NO EQUIPMENT INCLUDES PRINTS OR DOCUMENTATION (unless stated). NO CONNECTING CORDS OR CONNECTORS.

R-7

Microswitch Keyboard & Housing

We have a limited quantity of these used Micro-Switch keyboards. They have 50 keys, each individually wired (makes custom encoding easy), a 12 light status panel, and a panel of 2 or more status switches, all in an attractive steel sloped panel cabinet. The cabinet is grey, 13¾" W x 7½" deep x 3½" high. Each keytop is red, white or blue, has a ¾" square base and ½" dia. round top. The legends are on the base & can be scraped off & changed. Each keyswitch is SPST magnetic reed, N.O.

STOCK NO. C5444 Keyboard & housing, 4' cable Shipping weight 12 lbs. $22.50 each, 2/40.00

KEYTOPS & SWITCHES TO MAKE YOUR OWN KEYBOARD

We have a large selection of KEYTOPS and SWITCHES, made by RAYTHEON CO. The keytops come in black, grey and white, with contrasting legends. The switches mate with the tops, and are magnetic reed switches. The following combinations are available:

54 key typewriter set, keys only,	black	K9276	2.95
54 key typewriter set, keys only,	grey	K9278	2.95
54 key TTY set, no symbols	white	K9279	2.95
54 key TTY set, with symbols	white	K9282	2.95
54 key set, keys & switches	black	K9288	30.00
54 key set, keys & switches	grey	K9290	30.00
54 TTY set, no symbols keys & Sw.	White	K9291	30.00
54 TTY set, with symbols, keys & Sw.	white	K9291	30.00
11 Key Numeric set, Keys only	Black	K9283	1.50
11 Key Numeric set, Keys only	Grey	K9284	1.50
11 Key Numeric set, Keys only	White	K9295	1.50
12 Key numeric set, Keys only	white	K9286	1.50
11 Key Num.set, keys & switches	Black	K9293	7.00
11 Key Num. set, keys & switches	Grey	K9294	7.00
11 Key Num. set, keys & switches	White	K9295	7.00
12 Key Num. set, keys & switches	white	K9296	7.50
Blank key 1½ keys wide	white	K9297A	3/.25
Blank key 2 keys wide	white	K9297B	3/.25
K9297A with switch	white	K9298A	3/2.00
K9297B with switch	white	K9298B	3/2.00

VOLTAGE REGULATOR BOARDS

B5169 is a board containing 3 15 volt high current regulators with 0.1% regulation. 2 of the regulators are rated @ 3 Amps., and the other @ 6.0 amps. The current in each regulator may be doubled with the regulation going to 0.5%. All 3 regulators are short circuit proof, and 2 have electronic crowbar protection. Brand new, in factory boxes.

STOCK NO.B5169 $11.95 ea. 2/21.00

B9013 is a triple regulator with ± 12 volt regulation @ 200 ma. and the third regulator is a tracking regulator, providing regulation from 0 to 5 volts @ .5 A.

STOCK NO.B9013 $5.95 ea. 2/10.00

Both regulators above come with circuit diagrams.

16 KEY TOUCH TONE PAD

This 16 terminal TOUCH TONE pad was made for RAYTHEON. The keys are all SPST switches, brought out to 2 sets of terminals of 8 an; 9 pins, with .1 spacing so that they fir MOLEX IC terminals. The pad is 3" square. Individually packed with pin connection date. Ideal for touch tone uses. STOCK NO.K5488 $6.95 ea. 2/12.00

9PST DIP SWITCH

DIP packaged switch contains 9 SPST switches. Pin Spacing is 0.10", so it can be used with MOLEX terminals, or PC board. Ideal for computer use.

STOCK NO.K1009 $3.50 ea. 4/12.00

GOLD PLATED MOLEX TERMINALS

Gold plated MOLEX terminal strips in sections of 8 terminals. Any number of terminals may be obtained by cutting or adding sections.

STOCK NO. M7490 12 8 terminal pieces 1.49, 48/5.00, 96/8.00

7400 series TTL Parts Special

We have a series of surplus computer boards loaded with 7400 series TTL ICs, which are easily removed for reuse. We select 2 boards (samples shown) and guarantee at least 60 ICs, but most will have more than 70. The ICs are made mostly by T.I., and include gates, counters, flip flops, one shots, shift registers, decoders, etc. Less than 10 cents an IC, many of which sell for over $1. Shipping weight 1 lb. each.

STOCK NO. B5409 2 boards for $5.00, 5 boards for $10.00

TRANSFORMERS

PRIMARY	SECONDARY NO. 1	NO. 2	NO. 3	NO. 4	SIZE	LBS.	STK. NO.	PRICE
115v or 230v	34v, 3 A, CT	17v, 4 amps	11v, 5 amps	6.3v, 1.5 amps	3¾ x 4½ x 4¾	10	B9397	$12.95 ea, 2/24.00
115v	26v, 1 A, CT	6.3v, ½ amp			3 x 2½ x 2¼	4	B9907	3.75 ea, 2/7.00

DELTA ELECTRONICS CO.

P.O. BOX 2, AMESBURY MA 01913
Phone (617) 388—4705

MINIMUM ORDER $5.00. Include sufficient postage, excess refunded. Send for new 88 page Catalog 15, bigger than ever.

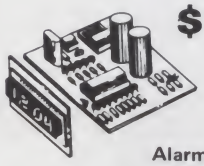
BANKAMERICARD and MASTERCHARGE now accepted, minimum charge $15.00. Please include all numbers. Phone orders accepted.

D-13

S.D. SALES CO.

P.O. BOX 28810 – K DALLAS, TEXAS 75228

JUMBO LED CAR CLOCK



$16.95 KIT

Alarm Option – $1.50
AC XFMR – $1.50

THE HOTTEST SELLING KIT WE EVER PRODUCED!

You requested it! Our first D.C. operated clock kit. Professionally engineered from scratch. Not a makeshift kluge as sold by others. Features:

A. Bowmar Jumbo –.5 inch LED array.
B. MOSTEK – 50250 – Super Clock Chip.
C. On board precision crystal time base.
D. 12 or 24 Hr. Real Time Format.
E. Perfect for cars, boats, vans, etc.
F. P.C. Board and all parts (less case) included.

50,000 SATISFIED CLOCK KIT CUSTOMERS CANNOT BE WRONG!

THIS MONTH'S SPECIALS

AMD – 8080A	$14.95
Z-80 CPU	49.95
82S129 1K PROM	2.50

1702A 2K EPROM

We tell it like it is. We could have said these were factory new, but here is the straight scoop. We bought a load of new computer gear that contained a quantity of 1702 A's in sockets. We carefully removed the parts, verified their quality, and are offering them on one heck of a deal. First come, first served. Satisfaction guaranteed! U.V. Eraseable. $6.95 ea. 4/$25

UP YOUR COMPUTER!

21L02-1 1K LOW POWER 500 NS STATIC RAM Time is of the essence!

And so is power. Not only are our RAM's faster than a speeding bullet but they are now very low power. We are pleased to offer prime new 21L02–1 low power and super fast RAM's. Allows you to STRETCH your power supply farther and at the same time keep the wait light off.

8 for $12.95

60 HZ CRYSTAL TIME BASE
S.D. SALES EXCLUSIVE!

$5.95 ea. 2/$10.00

KIT FEATURES:

A. 60HZ output with accuracy comparable to a digital watch.
B. Directly interfaces with all MOS clock chips.
C. Super low power consumption (1.5 MA typ.)
D. Uses latest MOS 17 stage divider IC.
E. Eliminates forever the problem of AC line glitches.
F. Perfect for cars, boats, campers, or even for portable clocks at ham field days.
G. Small size; can be used in existing enclosures.

Kit includes Crystal, Driver IC, PC board, plus all necessary parts and specs. At last count – over 20,000 sold!

S.D. SALES EXCLUSIVE

$12.95 MOS 6 DIGIT UP–DOWN COUNTER $12.95

40 PIN DIP. Everything you ever wanted in a counter chip. Features: Direct LED segment drive, single power supply (12 VDC TYPE.), six decades up/down, pre-loadable counter, separate pre-loadable compare register with compare output, BCD and seven segment outputs, internal scan oscillator, CMOS compatible, leading zero blanking. 1MHZ. count input frequency. Very limited quantity! WITH DATA SHEET

7400–19c	7411–29c	7451–19c	7490–65c	74153–75c
74LS00–49c	7413–50c	7453–19c	74LS90–95c	74154-1.00
7402–19c	7416–69c	7473–39c	7492–75c	74157–75c
74LS02–49c	7420–19c	7474–35c	7493–69c	74161–95c
7404–19c	7430–19c	74LS74-59c	7495–75c	74164-1.10
74L04–29c	7432–34c	7475–69c	7496–89c	74165-1.10
74S04–44c	7437–39c	7476–35c	74121-38c	74174–95c
74LS04-49c	7438–39c	7480–49c	74123-65c	74181-2.50
7406–29c	7440–19c	7483–95c	74132-1.70	74191-1.25
7408–19c	7447–85c	7485–95c	74S138-1.95	74192-1.25
7410–19c	7448–85c	7486–45c	74141-75c	74193-1.00

TTL INTEGRATED CIRCUITS 74195–69c

1000 MFD Filter Caps

Rated 35 WVDC Upright style with PC leads. Most popular value for hobbyists. Compare at up to $1.19 ea. from franchise type electronic parts stores. S.D. Special 4/$1.

Slide Switch Assortment

Our best seller. Includes miniature and standard sizes; single and multiposition units. All new, first name brand. Try one package and you'll reorder more! Special 12/$1.00

RESISTOR ASSORTMENT

¼W 5% & 10% PC leads. A good mix of values. 200/$2.

P.C. LEAD DIODES

1N4148/1N914 100/$2.00
1N4002-1A. 100 PIV 40/$1.

HEAVY DUTY Full Wave Bridge

25 AMP 50 PIV
$1.25

Disc Cap Assortment

PC leads. At least 10 different values. Includes .001, .01, .05, plus other standard values. 60/$1.00

$9.95 KIT

P.C. Board – 3.00
AC XFMR – 1.50

Do not confuse with Non-Alarm kits sold by our competition! Eliminate the hassle – avoid the 5314!

SIX DIGIT ALARM CLOCK KIT

We made a fantastic kit even better. Redesigned to take advantage of the latest advances in I.C. clock technology. Features: Litronix Dual ½" displays, Mostek 50250 super clock chip, single I.C. segment driver, SCR digit drivers. Greatly simplified construction. More reliable and easier to build. Kit includes all necessary parts (except case). P.C.B. or XFMR optional.

NEW! WITH JUMBO LED READOUTS!

Motorola SCR

2N4443. 8 AMP 400 PIV. P.C. Leads 3/$1.

FAIRCHILD - TBA 641

4W. Audio power Amp. Just out! In special heat sink DIP. One super audio IC. $1.50 with data

FND-359 -Led Readout

.4 IN. Common Cathode. High effeciency. Has FND-70 PIN OUT. 59c

OUR CATALOG

is chocked full of rare parts bargains, deals, RAM or CPU kits, plus much more. Yours FREE!

PRICES SHOWN SUBJECT TO CHANGE WITHOUT NOTICE.

$15.95

COMPUTER POWER SUPPLY

A very fortunate purchase. One of the best industrial quality REGULATED supplies we have seen. High performance, small size. Input is 120 VAC 60 HZ. Has the following regulated outputs: **–5VDC@800MA; –15VDC @ 1.25 AMP; –25VDC @ 180 MA.** Sold at a fraction of original cost. Do yourself a favor and order NOW. We expect a quick sellout.

NEXT MONTH:
S.D. will have music for your ears. Watch our ads.

For your Imsai or Altair 8080 Computer:
Z–80 CPU Kit – $149. 4K Low Power Ram Kit – $89.95

Terms: Money back guarantee. No COD. Texas residents add 5% sales tax. Add 5% of order for postage & handling. Orders under $10. add 75c. Foreign orders: US funds only!

Call your Bankamericard or Master Charge order in on our continental United States toll free Watts:

1–800–527-3460

Texas Residents Call Collect:
214/271-0022

Special Thanks to:
Dennis, Fred, Abe, Bill, Sam, Hal, Tom, Alex, John, Ely, and Larry

S.D. SALES CO.
P. O. BOX 28810 K
Dallas, Texas 75228

ORDERS OVER $15.00 – CHOOSE $1.00 FREE MERCHANDISE

computer display terminal

This display terminal has an integral controller B/W cathode ray rube and keyboard. The system has a serial I/O interface for communication and I/O interface for a printer.

External logic & power pack not shown.

DISPLAY (P/N 4802-1095-501) FEATURES:
- 17" B/W CRT
- 41 lines of data
- 52 characters per line
- Characters are generated by a diode matrix "graphic" technique
- 21 special push-buttons wired for a program call up
- Brightness Control
- Self-contained power supply

KEYBOARD (P/N 4802-1115-501) FEATURES:
- Reed switch technology
- 54 data keys
- 28 special keys detachable with cable

LOGIC UNIT (P/N 4802-1157-502) FEATURES:
- 1024 by 6 bit core memory
- Printer I/O interface
- Communication I/O interface

POWER: 115V, 50/60 Hz, 500 Watts

WEIGHT: 210 lbs. (including logic unit, keyboard, display and cables.)

$ 180.00

4 way cursor control, graphics display.

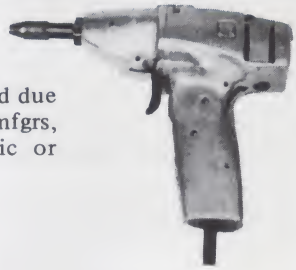
The story: These are unused terminals made for airport ticketing & seat assignment. After several years of storage they require tinkering to make operable. We have some hints printed such as cleaning PC fingers. One of our customers has this tied into his KIM-1, another has his running with his IMSAI. We have data on this. Should be useable on most common computers. A hell of a deal and all for a paltry $180.00. Don't be left out as many were on our past VIATRON deal. Sold "as is" all sales final.

FOB LYNN MASS (you pay shipping)
Check with order please.

WITH COMPLETE DOCUMENTATION

Here is a real deal in a PC module 6x5 sockets (30). List price over $50 each, most by AUGAT some pre-wired. New, unused boxed, 14 or 16 pins 5x6 sockets. $15.00 each or 2 for $25, state your choice 14 or 16 pin.

WIRE WRAP GUNS



Used wire wrap guns, released due to factory closure. Various mfgrs, some Ingersol Rand. electric or air.
No collets. State choice.
Cost over $100.00 each.

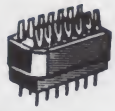
Our price only $15.00 each.

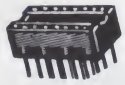
SOLAR CELLS

Designed for the space program, these are the highly efficient silicon high output cells. Used for powering equipment, charging batteries. Made by Ion Physics Corp. Each with spec sheet.
Size .394 x .788" 65 mA, .43 V
$1.25 12/$12.00
Size .788 x .788" 125 mA, .43 V
$1.60 12/$15.00

shown actual size

SILICON SOLAR CELL





IC Sockets, while they last . . .

8 Pin	10/$1.00
14 Pin	10/$1.25
16 Pin	10/$1.50
18 Pin	10/$1.75
14 Pin IC connector	10/$1.25

Meshna

Please add shipping cost on above. Minimum order $10

FREE CATALOG SP-9 NOW READY

P.O. Box 62K, E. Lynn, Massachusetts 01904 M-2

The Compucolor 8001 System.

It's A Stand Alone Micro Computer With Color Input/Output Capabilities All In One Package. For Only $2995.

If you're looking for an input device, an output device and a micro computer all in one package, you've found it. The Compucolor 8001. It's here now, in color, on sale for only $2995.

We gave it a memory of its own.

And Floppy Tape Memory is just for starters. Look at these other features. BASIC Language, 8080 CPU, 8 color CRT Terminal, 8K RAM Workspace, Selectable Baud Rate to 9600, Two RS 232 I/O's, Keyboard with 16 Function Keys, Background Color, Lower Case ASCII Characters, Roll, Insert/Delete, 48 Line X 80 Characters/Line, 2X Character Height, thorough operating instructions and a Graphics Mode with 160 X 192 Elements. And our unique Nine Sector Convergence System guarantees you quick set-up, exceptional stability and outstanding color registration in three to five minutes. If you can find a better buy in a color Intelligent CRT and Micro Computer system, let us know. We think we've got the best of both worlds at the best price going. And we want to prove it to you.

Name your game.

After all, you'll have your very own personal computer right at your fingertips. For the most simple or complex tasks. Or just plain fun. The applications are unlimited. Color graphics and computations, check book balancing, educational instruction, tutoring and a unique variety of computer games. Like Star Trek and Hangman and Pong. You can even sit back and enjoy a game of chess. Like we said, the applications are unlimited.

How about a little demonstration?

You'll find a list of our distributors at the bottom of the page. So drop by and ask for a demonstration. Get some answers to your questions. And if you aren't near one of our distributors, give us a call. We've got the answers. The Compucolor 8001. You won't find a better buy in a color CRT Terminal and Micro Computer.

Compucolor Corporation
A subsidiary of
Intelligent Systems Corp.®

5965 Peachtree Corners East
Norcross, Georgia 30071
Telephone (404) 449-5961

CALIFORNIA
Byte Shop
155 Blossom Hill Rd.
San Jose, Ca. 95123
Larry Grihalva
(408) 226-8383

Computer Store
1093 Mission St.
San Francisco, Ca. 94103
Al Chern
(415) 431-0640

CALIFORNIA
The Computer Center
8205 Ronson Rd.
San Diego, Ca. 92111
Ron Eate
(714) 292-5302

The Computer Mart of Los Angeles
625 W. Katella No. 10
Orange, Ca. 92667
George Tate
(714) 633-1222

GEORGIA
The Computer Systems Center
3330 Piedmont Rd., NE
Atlanta, Ga. 30305
Jim Dunion
(404) 231-1691

ILLINOIS
Itty Bitty Machine
1316 Chicago Ave.
Evanston, Ill. 60201
Jim Bannish
(312) 328-6800

INDIANA
Home Computer Shop
10447 Chris Dr.
Indianapolis, Ind. 46229
James B. Baughn
(317) 894-3319

MASSACHUSETTS
The Computer Store
120 Cambridge St.
Burlington, Mass. 01803
Sid Halligan
(617) 272-8770

WASHINGTON
Retail Computer Store
410 N.E. 72nd Street
Seattle, Wash. 98115
Tim Broom
(206) 524-4101

Or Contact Us Direct

5965 Peachtree Corners East
Norcross, Georgia 30071
Telephone (404) 449-5961

kilobaud

index of advertisers

A-29 Ace Electronic Parts 127
A-2 Aldelco 87
A-28 Apple Computer Co. 61
C-5 Communications Electronics 111 & 123
C-35 Computer Components 81
C-28 The Computer Corner 115
C-33 Computer Enterprises 144
C-30 Computer Mart of NJ 73
C-32 Computer Mart of NY 138
C-34 The Computer Store, Inc. 49
C-27 Computer Warehouse Store 129
C-31 Computer Workshop 115
C-9 Continental Specialties 1
D-13 Delta Electronics 141
D-12 Digital Group 27
E-13 ECD Corp. C III
G-4 Godbout 119
H-10 Hufco 109
I-5 Imsai 33
I-6 Intelligent Systems 144 & Card Insert
I-9 Integrated Circuits 137
I-10 International Data Systems 127
J-1 James Electronics 133
L-3 Logical Services 73
M-2 Meshna 143
M-14 Micro Peripherals 63
M-12 Micro Software Specialists 87
M-6 Mini Micro Mart 99
M-4 MITS C IV, 62 & 63
M-7 Morrow's Micro-Stuff 119
N-1 National Multiplex 77
N-7 Newman Computer Exchange 73
O-1 Ohio Scientific 47
O-5 OK Machine & Tool 53
O-3 Optoelectronics 135
P-9 Pais Electronics 67
P-12 Peripheral Vision 95
P-8 Personal Computing Co. 125
P-13 Prime Radix 26
P-11 Proko Electronics Shoppe 94
Q-4 Quality Security Systems 115
R-12 Rainbow Computing Inc. 138
R-7 Rondure Co. 140
S-1 Scelbi 87
S-26 Scientific Research 38 & 39
S-2 S. D. Sales Co. 142
S-22 Seals Electronics 25
S-6 Southwest Technical Products C II
S-28 Sylvanhills Lab Inc. 144
T-11 Tarbell Electronics 108
T-8 Technical Design Lab 5
T-12 Technical Systems Consultants 7
T-13 Teletypewriter Comm. Spec. 80
T-1 Tri-Tek 139
W-13 Wasatch Semiconductor Products 108
W-5 Wave Mate 9 & 11
W-16 World Wide Electronics 126
W-14 WWW Enterprises 63

From *Kilobaud*
73 Subs 37
KB Booknook 42 & 43
KB Subs 48
KB Comp list 87
KB Hobby Computers 89
KB Wats line 127
KB Back issues 127

PETERBOROUGH NH 03458

kilobaud reader service

Circle appropriate Reader Service # for desired company brochures, data sheets or catalogs and mail to Kilobaud Magazine, Peterborough NH 03458. Include your zip code, please. Send money directly to advertisers. LIMIT: 25 requests.

A-29	C-32	G-4	M-2	O-1	Q-4	S-28	W-16
A-2	C-34	H-10	M-14	O-5	R-12	T-11	W-14
A-28	C-27	I-5	M-12	O-3	R-7	T-8	
C-5	C-31	I-6	M-6	P-9	S-1	T-12	KILOBAUD
C-35	C-9	I-9	M-4	P-12	S-26	T-13	
C-28	D-13	I-10	M-7	P-8	S-2	T-1	
C-33	D-12	J-1	N-1	P-13	S-22	W-13	
C-30	E-13	L-3	N-7	P-11	S-6	W-5	

☐ Subscriber ☐ Newsstand buyer

PLEASE PRINT OR TYPE

Name _____

Address _____

City _____ State _____ Zip _____

Coupon Expires in 60 Days K 4/77

PLACE STAMP HERE

kilobaud

PETERBOROUGH NH 03458

Att: Mail Order

I want to subscribe!

Send me Kilobaud for:

☐ 1 year — $15.00* ☐ Renewal

☐ 3 years — $36.00* ☐ New sub.

☐ Life Subscription — $155.00

Enclosed $_____ ☐ Cash ☐ Check ☐ Money Order

☐ American Express ☐ BankAmericard ☐ Master Charge

Credit card #_____ Interbank # _____

Expiration date _____ Signature _____

☐ Bill me directly Signature _____

Name _____

Address _____

City _____ State _____ Zip _____

*U.S. & Canada ONLY. Others write for foreign rates. K4/77

FIRST CLASS
Permit No. 1024
Peterborough NH 03458

BUSINESS REPLY MAIL

NO POSTAGE NECESSARY IF MAILED IN UNITED STATES

POSTAGE WILL BE PAID BY

kilobaud

PETERBOROUGH NH 03458

Att: Reader Service

PLEASE PRINT OR TYPE Please send me the following Kilobaud products:

Quantity	Description	Unit Price	Total

NOTE: $1.00 Handling charge for orders less than $4.00.

Total _____
Handling Charge _____
Order Total _____

Enclosed $_____ ☐ Cash ☐ Check ☐ Money Order ☐ C.O.D.
Bill: ☐ American Express ☐ BankAmericard ☐ Master Charge
Credit Card # _____ Interbank #_____
Expiration date _____ Signature _____
Name _____
Address _____
City_____ State _____ Zip _____ K 4/77

PLACE STAMP HERE

kilobaud

PETERBOROUGH NH 03458

Att: Subscriptions

books,etc.

Microcomputer Dictionary $15.95
Computer Programming Handbook $8.95
My Computer Likes Me . . . When I Speak BASIC $2.
Scelbi's Galaxy Game for the "8008"/"8080" $14.95
6800 Software Gourmet Guide & Cookbook $9.95
CMOS Cookbook $9.95
Hobby Computers Are Here $4.95
Scelbi's First Book of Computer Games $14.95
The Story of Computers $4.95
Microcomputer Primer $7.59
The New Hobby Computers $4.95
Test Equipment Library $4.95 Each
- Vol. I Component Testers
- Vol. II Audio Frequency Testers
- Vol. III Radio Frequency Testers

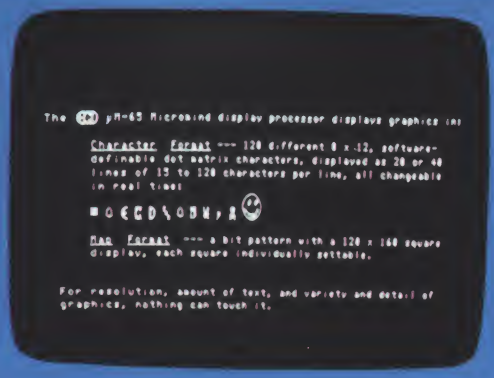
Novice Study Guide $4.95
General Class Study Guide $5.95
VHF Antenna Handbook $4.95
Weather Satellite Handbook $4.95
SSTV Handbook $5.00
RF and Digital Test Equipment You Can Build $5.95
What To Do After You Hit Return $6.95
101 Games in BASIC $7.50
BASIC $4.95
TVT Cookbook $9.95
TTL Cookbook $8.95
73 Back Issues $2.50 Each

kilobaud

moving?

Mail us your new address as soon as possible to insure uninterrupted delivery of your magazines. Allow EIGHT Weeks Advance Notice.
You may use the pre-printed order card for your change of address:

1. Write in "description" column "MY OLD ADDRESS WAS:"
2. Attach old label if available . . . otherwise print *clearly* the old address (don't forget the zip code).
3. Clearly print your name & new address in the spaces provided at the bottom of the card.



Key Into Maxi-Power @ Micro-Price

Micromind is an incredibly flexible, complete and expandable, hardware/software, general purpose computer system. You won't outgrow it.

Hardware includes an 80 key, software-definable keyboard, I/O interface board, 6500A-series microprocessor (powerful enough for advanced computing), a high-detail graphics and character display processor, power supply, rf modulator, and connections for up to 4 tape recorders plus TV or monitor. An interconnect bus

permits 15 additional microprocessors, parallel processing and vastly increased computing power.

System software—including ECD's own notsoBASIC high level language, on *advanced error-correcting* tape cassettes—provides a word processing editor, a powerful assembler, a debugger, a file system, graphic routines, and peripheral handlers. We also include dynamic graphic games: Animated Spacewar and Life.

ECD's standard Micromind μM-65 supplies 8K bytes of memory. Additional 32K byte expansion boards and a mapping option give Micromind expandable access to 64 Megabytes. Utilizing software-controlled I/O channels, Micromind's advanced encoding techniques load data from ordinary tape recorders at 3200 bits per second.

Micromind comes to you ready-to-use, factory assembled and fully tested. Among microcomputers, it has the largest memory capacity and the fastest storage. You're looking at the work of the finest display processor on the market. You won't find a microcomputer with a more powerful CPU. You won't find a computer with a more flexible keyboard. You won't find anything to touch it at **$987.54.**

So, quit the kluge scene and key into Micromind. You'll be a main frame performer, with all the comforts of home. We're not fooling...this is the cat's μ!

ECD CORP.
196 Broadway, Cambridge, Mass. 02139
(617) 661-4400

ECD

Name _____

Address _____

City/State _____ Zip _____

☐ Fantastic! Check enclosed: **$987.54.** Shipping paid by ECD
☐ BankAmericard ☐ Master Charge Mass. Resident add 5% Sales Tax

_____ Expiration Date _____

Signature _____

☐ Send me your brochure.

7

Actual unretouched photographs.

Now you can buy an Altair™8800b or an Altair 680b computer right off the shelf. Altair plug-in boards, peripherals, software and manuals are also available. Check the list below for the MITS dealer in your area.

off the shelf.

ALTAIR COMPUTER CENTER
8105 SW Nimbus Ave.
BEAVERTON, OR 97005

COMPUTER KITS (S.F. area)
1044 University Ave.
BERKELEY, CA 94710
(415)-845-5300

THE COMPUTER STORE
(Arrowhead Computer Co.)
820 Broadway
SANTA MONICA, CA 90401
(213)-451-0713

GATEWAY ELECTRONICS, INC.
OF COLORADO
2839 W. 44th Ave.
DENVER, CO 80211
(303)-458-5444

COMPUTER SHACK
3120 San Mateo N.E.
ALBUQUERQUE, NM 87110
(505)-883-8282; 883-8283

ALTAIR COMPUTER CENTER
4941 East 29th St.
TUCSON, AZ 85711
(602)-748-7363

ALTAIR COMPUTER CENTER
611 N. 27th St. Suite 9
LINCOLN, NB 68503
(402) 474-2800

COMPUTER PRODUCTS UNLIMITED
2412 Broadway
LITTLE ROCK, AR 72206
(501)-371-0449

ALTAIR COMPUTER CENTER
110 The Annex
5345 East Forty First St.
TULSA, OK 74135
(918)-664-4564

ALTAIR COMPUTER CENTER
5750 Bintliff Drive
HOUSTON, TX 77036
(713)-780-8981

COMPUTERS-TO-GO
4503 West Broad St.
RICHMOND, VA 23230
(804)-335-5773

MICROSYSTEMS (Washington, D.C.)
6605A Backlick Rd.
SPRINGFIELD, VA 22150
(703)-569-1110

THE COMPUTER STORE
Suite 5
Municipal Parking Building
CHARLESTON, W. VA. 25301
(304)-345-1360

THE COMPUTER ROOM
3938 Beau D'Rue Drive
EAGAN, MN 55122
(612)-452-2567

THE COMPUTER STORE
OF ANN ARBOR
310 East Washington Street
ANN ARBOR, MI 48104
(313)-995-7616

THE COMPUTER STORE, INC.
(Hartford area)
63 South Main Street
WINDSOR LOCKS, CT 06096
(203)-627-0188

CHICAGO COMPUTER STORE
517 Talcott Rd.
PARK RIDGE, IL 60068
(312)-823-2388

GATEWAY ELECTRONICS, INC.
8123-25 Page Blvd.
ST. LOUIS, MO 63130
(314)-427-6116

BYTE'TRONICS
Suite 103
1600 Hayes St.
NASHVILLE, TN 37203
(615)-329-1979

THE COMPUTER STORE, INC.
120 Cambridge St.
BURLINGTON, MA 01803
(617)-272-8770

ALTAIR COMPUTER CENTER
269 Osborne Road
ALBANY, NY 12211
(518)-459-6140

THE COMPUTER STORE
OF NEW YORK
55 West 39th St.
NEW YORK, NY 10018
(212)-221-1404

THE COMPUTER SYSTEMCENTE
3330 Piedmont Road
ATLANTA, GA 30305
(404)-231-1691

MARSH DATA SYSTEMS
5405 B Southern Comfort Blvd.
TAMPA, FL 33614
(813)-886-9890

mits